MASTER EN SCIENCES INFORMATIQUES

Informatisation des schémas économiques

Tasiaux, B.

Award date:
1984

Awarding institution:
Universite de Namur

FACULTÉS NOTRE-DAME DE LA PAIX

NAMUR

INFORMATISATION

DES SCHEMAS ECONOMIQUES

plus spécifique à progr. retour ?
déjà généralité p.80
p.88 légende Schéma complet

Directeurs : R.P. BERLEUR

ET J-C JACQUEMIN

Mémoire présenté pour
l'obtention du diplôme
de Licencié et Maître
en Informatique

par b. Tasiaux

Année académique 1983-1984

Je voudrais remercier ici Monsieur J.C. Jacquemin pour sa direction éclairée, sa gentillesse et sa disponibilité lors de l'élaboration de ce mémoire.

Que Monsieur B. Lecharlier trouve ici l'expression de ma gratitude pour son aide précieuse lors de la conception de la partie mathématique de ce mémoire.

Je voudrais également exprimer ma reconnaissance à Marc Ervier pour ses conseils judicieux et amicaux.

Enfin, merci à toute personne, étudiant, assistant, ami ou parent qui, de près ou de loin, m'ont aidée à réaliser ce mémoire.

TABLE DES MATIERES

INTRODUCTION

INTRODUCTION

A. Exemple

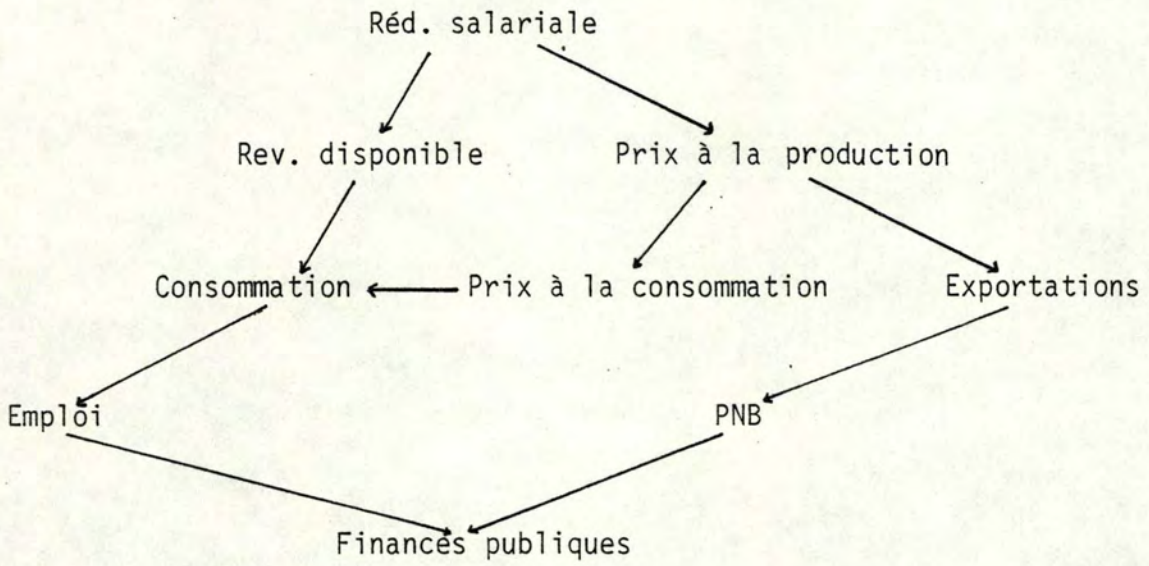
Parmi les nombreuses tentatives pour améliorer les finances publiques, le gouvernement Martens V a établi un schéma de modération salariale. Sans en connaître tous les détails et tous les mécanismes, on peut, cependant, avec un peu de bon sens, tâcher de voir quels seront les effets d'une telle décision sur l'économie belge.

A priori, diminuer les revenus revient à diminuer le revenu disponible de chaque ménage. Dès lors, ceux-ci dépenseront moins.
On peut donc imaginer une diminution de la consommation qui, a son tour, provoquera une diminution de l'emploi.Mais une augmentation du chômage se traduit par une augmentation de la dette publique.
On peut, voyant cela, se poser des questions quant au bien-fondé d'une telle décision dans le cadre d'une amélioration des finances publiques.

Mais, d'un autre point de vue, la diminution des salaires doit avoir un impact sur les prix à la production, et donc sur les prix à la consommation. Les ménages pourraient, dès lors, acheter plus puisqu'ils achèteraient moins cher.
Dans quelle mesure cette augmentation de la consommation va-t-elle pallier la baisse prévue précédemment ?

D'autre part encore, on imagine que si les prix diminuent, on va exporter plus. Il y aura donc une variation de la balance commerciale. Celle-ci influançant le PNB, qui lui-même affectera les finances publiques.
Pour avoir une idée plus précise des choses, il faudrait pouvoir évaluer le temps qui s'écoule entre l'entrée en application de la réduction salariale et l'enregistrement de ses différentes conséquences.
Quels sont les effets à court, moyen ou long terme ?

La hausse de la consommation va-t-elle survenir avant, après ou en même temps que la baisse de consommation également prévue ?
Il faudrait aussi pouvoir mesurer différents effets. Quelle sera l'influence la plus forte, la baisse ou la hausse de consommation ? Quelles sont les sommes en jeu ?
Un petit dessin valant mieux qu'un long texte, on peut représenter toutes ces interactions en dessinant des points pour représenter les différents postes de l'économie belge touchés par cette réduction des salaires.
On ajoute des flèches entre ces points pour indiquer l'influence qu'un poste peut avoir sur l'autre. Ainsi obtient-on une ébauche de ce que l'on appellera les schémas économiques.



B. Définition du schéma économique.

Les économistes disposent de modèles qui permettent de mettre en évidence les relations qui existent entre différents centres de variation. Une visualisation de ces modèles consiste en une représentation graphique au sens mathématique du terme.
Un graphe est, en effet, un couple (X, U) où X est un ensemble de points appelés sommets du graphe, et où U est un sous-ensemble de XxX tel que

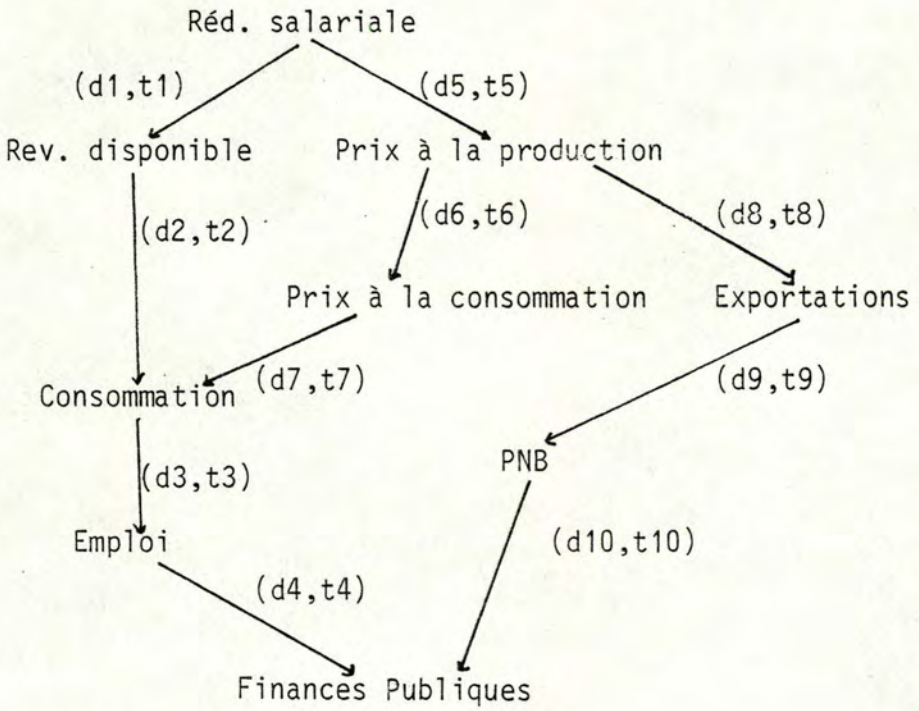
$u = (x, y)$ appartenant à U représente un arc reliant x à y.

Un schéma économique prend les centres de variation des modèles économiques comme ensemble des sommets du graphe, et représente l'influence qu'un centre de variation cvi peut avoir sur un autre centre de variation cvj, par un arc allant de cvi à cvj.
On notera dorénavant cv pour centre de variation.
A chacun des arcs, le schéma économique associe deux mesures :

1. le délai : le temps nécessaire à la variation du centre origine de l'arc pour provoquer une variation du centre extrémité de l'arc.

2. un coéfficient, ou mesure de l'influence que l'arc représente.

Ainsi, le dessin représentant les différents effets d'une réduction salariale est-il complété pour en faire un schéma économique.



Ainsi, la diminution de la consommation pourra-t-elle être mesurée et vaudra :

t1 x t2 x réduction salariale = diminution consommation.

L'augmentation de la consommation vaudra, quant à elle :

t5 x t6 x t7 x réduction salariale

On est maintenant plus à même de comparer ces deux grandeurs.

Et, grâce aux délais introduits dans ces schémas, on pourra se faire une idée des effets de la réduction salariale à court, moyen ou long terme.

C. Avantages et inconvénients de la représentation.

On perçoit bien l'utilité de tels schémas : utiles passivement d'abord, pour comprendre les tenants et aboutissants de certaines décisions politiques ou de divers faits économiques enregistrés, et utiles activement ensuite en tant qu'outil didactique d'élaboration de modèles économiques et de simulation. Ces schémas permettent, en effet, d'imaginer et d'essayer des propositions de modèles différents.

Ces schémas permettent également de simuler différents scénarios à un coût minimal, et de choisir le(s) meilleur(s) de ces scénarios pour les analyser de façon détaillée sur de plus gros systèmes tels que MARIBEL ou SERENA pour lesquels une seule itération coûte cher.

D'autre part, il peut exister tellement de centres de variation impliqués dans de tels schémas qu'on imagine facilement les difficultés que quelqu'un rencontrera à mettre de l'ordre dans tout cela, avec son seul bon sens.
Comment va-t-il repérer facilement tous les sommets influencés par une variation d'un cv quelconque ?
Comment repérer les cv influencés à court, moyen ou long terme, sans risquer de s'embrouiller et sans risquer d'en oublier ?
Comment calculer la mesure des influences d'une variation d'un cv sur les autres centres sans erreur ?
D'où l'utilité d'informatiser. Pour ne pas perdre son temps à essayer de voir clair dans un schéma d'autant plus difficile à manier que les renseignements sont nombreux.
Et puisque les réponses aux questions posées ne demandent que de la manipulation de renseignements, pourquoi ne pas laisser faire la machine, plus rapide et moins sujette aux erreurs, pour mieux nous intéresser à l'élaboration de nouvelles solutions.

D. De l'informatisation des schémas économiques.

Pour aider à la manipulation des nombreuses données attachées à un schéma économique, on envisage d'informatiser de tels schémas, ainsi que les calculs qui s'y rattachent.
Qu'est-ce à dire ?

On se place dans un schéma économique donné. On fournit une impulsion à l'un quelconque des centres de variation du schéma, que l'on appellera centre de variation de départ, noté cvd.

Le but du mémoire est de construire un programme qui procure à l'utilisateur une liste claire des centres de variation influencés par une variation du cvd après un court, moyen ou long terme.
Ou, plus exactement, la liste des cv influencés dans un délai total inférieur ou égal à un temps imparti noté TI.

Le programme donnera également la mesure des influences qui auront été relevées.

D'autre part, et ceci est important, on s'efforcera de mettre à la disposition de l'utilisateur les moyens nécessaires à l'élaboration de différents schémas.

Mais un tel but ne s'atteint pas sans problèmes.

Problèmes d'ordre mathématique d'abord, car les schémas économiques ne sont pas réellement des graphes quelconques. Qu'en est-il des circuits que l'on rencontre ? Spécialement si ceux-ci sont constitués d'arcs représentant des influences de délai nul ? Peut-on imaginer des boucles ?
D'autre part, il ne sera pas aisé de construire un algorithme réalisant l'objectif fixé et de prouver sa correction.

Des problèmes d'ordre informatique se posent également, et ce à deux niveaux.

Au niveau de l'utilisateur, comment faire "coller" la représentation informatique des schémas économiques à ceux-ci ? Ceci est d'autant plus difficile que les schémas économiques n'ont, en fait, aucune règle de syntaxe précise quant à leur construction et leur représentation. De plus, on se rend aisément compte que le succès du logiciel sera proportionnel à sa facilité d'usage et sa convivialité vis-à-vis de l'utilisateur, qualités indispensables à tout bon outil pédagogique.

Au niveau des performances du programme, comment garder un maximum d'efficacité tout en restant portable ? Quelle structure de données adopter dans cette optique ?

Une fois le programme construit, on l'utilisera pour étudier, en détail, les effets sur l'économie belge d'une réduction des salaires réels, appliquée selon le plan de modération salariale conçu par le gouvernement Martens V.

* * * * * * * *

PREMIERE PARTIE

UN ALGORITHME DE SOLUTION

A. RAPPEL D'ELEMENTS DE LA THEORIE DES GRAPHES.

Intuitivement, un graphe est donc un schéma constitué par un ensemble de points en nombre fini ou dénombrable qui sont appelés sommets, reliés entre eux par des branches orientées ou non. On dit que le graphe est un graphe orienté lorsque toutes les branches sont orientées.
Ce qui s'applique aux schémas économiques. Dans ce cas, les branches sont appelées les arcs du graphe.

> Un chemin est une suite C = (a1, a2, ..., aN) d'arcs telle que l'extrémité terminale d'un arc aI coïncide avec l'extrémité initiale de l'arc suivant aI + 1. Ceci pour i de 1 a N-1.

Le nombre N d'arcs qui composent la suite C est la longueur du chemin C.

L'extrémité initiale de a1, soit cv1, par exemple, et l'extrémité terminale de aN, soit cvN, sont respectivement appelés origine et extrémité finale du chemin C, et on dit que le chemin C joint cv1 à cvN.
Si cv1 égale cvN, alors, on dit que C est un circuit.

- Le sommet cvi est un ascendant du sommet cvj, lorsqu'il existe un chemin ayant cvi et cvj comme origine, et extrémité terminale respectivement.

- Le sommet cvj est un descendant du sommet cvi, lorsque cvi est un ascendant du sommet cvj.

- La longueur du plus court chemin joignant cvi à cvj est l'écart du sommet cvi au sommet cvj.

- Le sommet cvj est un descendant d'ordre k du sommet cvi, si cvj est un descendant de cvi et si l'écart de cvi à cvj est égal à k.
Dans ce cas, on dit aussi que cvi est un ascendant d'ordre k de cvj.

- Le poids total d'un chemin C est la somme des poids de chacun des arcs qui composent le chemin

$$p(C) = \sum p(a) \text{ pour tout a appartenant à C.}$$

B. CARACTERISATION DES GRAPHES REPRESENTANT DES SCHEMAS ECONOMIQUES.

On a déclaré précédemment que les schémas économiques avaient l'ensemble des centres de variation des modèles économiques étudiés pour ensemble des sommets et que l'arc (cvi, cvj) représente l'influence qu'une variation de cvi peut avoir sur cvj.
Chacun des arcs est caractérisé par deux poids :

- le délai : temps nécessaire à la variation de centre origine de l'arc pour provoquer une variation du cv extrémité de l'arc.
- le coéfficient : mesure de l'influence représentée par l'arc;

On suppose, de plus, que ces graphes ont les caractéristiques suivantes :

- Il s'agit d'1-graphe. C'est-à-dire, il existe au plus un arc reliant un sommet quelconque à un autre sommet quelconque du graphe.
- Le graphe ne contient pas de circuit de délai total nul. La raison de cette hypothèse sera expliquée et justifiée par la suite.
- Le graphe ne contient pas de boucle. Car un cv ne s'influence pas lui-même, sauf,peut-être, avec un délai nul. Mais on aurait alors un circuit de longueur 1, et de délai total nul, ce qui contredirait la seconde hypothèse.

C. IDEE DU PETIT POUCET.

L'algorithme du Petit Poucet est, au départ, un algorithme de visite des sommets du graphe. On procède comme suit :

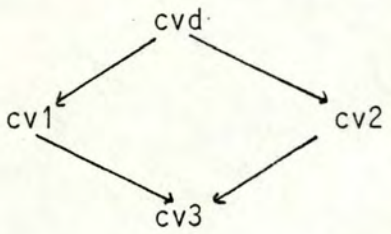
soit à visiter tous les sommets d'un graphe; l'une des façons les plus simples de faire ces visites consiste à procéder ainsi:

- ayant visité le sommet cvj, se déplacer sur un arc (cvj, cvk) incident à cvj.

- si l'on a déjà visité le sommet cvk, revenir en cvj et choisir un autre sommet incident à cvj;
  sinon, visiter le sommet cvk, et appliquer le processus ainsi défini de manière récursive à cvk;
- si tous les arcs incidents à cvj ont été explorés, revenir au sommet qui a permis d'atteindre cvj, soit cvi, et continuer à explorer tous les arcs incidents à cvi vers l'extérieur.

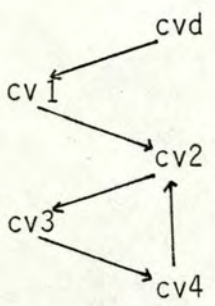
On voudrait adapter l'algorithme du Petit Poucet au problème des schémas économiques. On va donc considérer un algorithme qui permettrait de visiter tous les sommets du graphe, descendants du cvd. Et pour chacun de ces sommets, on vérifierait que le délai total du chemin emprunté menant du cvd à ce sommet ne dépasse pas le temps imparti.

Mais cette idée se heurte à deux caractéristiques inhérentes aux schémas économiques.
Premièrement, pour un schéma comme celui-ci, si on procède comme indiqué,



on passera du cvd à cv1, puis de cv1 à cv3. Comme cv3 n'a pas de descendant, on remonte à cv1 puis à cvd. On repart alors sur cv2. Mais on ne pourra pas atteindre cv3 puisqu'il aura déjà été visité. Mais alors, on néglige l'influence que représente l'arc (cv2, cv3) !

Secondement, que se passe-t-il pour un graphe tel que celui-ci ?



Supposons que l'on dépasse le temps imparti en cv2, il faut, selon le principe du Petit Poucet, remonter au sommet qui permit d'atteindre cv2. Comment savoir s'il s'agit de cv4 ou de cv1 ? Comment retenir que l'on a déjà fait un ou plusieurs tours de boucle (cv2, cv3, cv4) ?

L'adaptation faite de l'algorithme du Petit Poucet n'est donc pas une solution adéquate au problème des schémas économiques, puisqu'il arrive fréquemment d'y trouver de telles configurations d'arcs.

D. IDEE DE LA RECURRENCE.

1. Présentation de l'idée.

Notons cvd-chemin, tout chemin dont l'origine est le cvd.

L'idée est de découvrir tous les sommets extrémités de cvd-chemins de délai total I, connaissant déjà les sommets extrémités de cvd-chemins de délai I - 1, I - 2, ... 0.

Pour réaliser cela, on imagine une matrice D (0:TI, 1:NC) où NC représente le nombre de colonnes de la matrice, que l'on tentera de déterminer par la suite.
Chaque ligne I de la matrice D représente tous les sommets extrémités de cvd-chemins de délai total I.

Un sommet sera représenté sur cette ligne autant de fois qu'il est extrémité de cvd-chemins différents de délai total I. On considère que deux cvd-chemins sont différents si dans la suite des arcs qui composent un des cvd-chemins, il existe au moins un arc qui ne soit pas repris dans la suite des arcs qui composent l'autre cvd-chemin.

Chaque ligne de la matrice D sera constituée à partir des lignes précédentes.

Ainsi, au départ la ligne 0 comprend-elle le cvd, ainsi que tous les sommets descendants du cvd par un cvd-chemin de délai total nul.

Soit DES(cvd) l'ensemble des descendants du cvd,
soit cvi appartenant à DES(cvd)
si le délai total du cvd-chemin joignant cvd à cvi est nul,

alors cvi sera sur la ligne 0 de la matrice D
sinon cvi ne s'y trouvera pas !

Pour construire la Ième ligne à partir des lignes précédentes de la matrice, on opèrera comme suit :

Un élément cvj appartient à la ligne I de la matrice D
ssi il existe au moins un arc
dont l'extrémité soit cvj
dont l'origine soit un élément cvk
d'une ligne k de la matrice, avec k compris entre 0 et I
dont le délai soit de I - k.

Notons S(cv), l'ensemble des suivants d'un sommet cv quelconque
On fera :
Pour tout cvi appartenant à la 0ème ligne,
faire pour tout cvj appartenant à S(cvi)

si d(cvi, cvj) = I
alors cvj appartiendra à la ligne I

Pour tout cvi appartenant à la 1ère ligne,
faire pour tout cvj appartenant à S(cvi),
si d(cvi, cvj) = I - 1
alors cvj appartiendra à la ligne I,

.
.
.

Pour tout cvi appartenant à la I - 1ème ligne,
faire pour tout cvj appartenant à S(cvi),
si d(cvi, cvj) = 1
alors cvj appartiendra à la ligne I.

Afin que cette Ième ligne soit complète, il faut encore ajouter à ces sommets les sommets descendants, avec un délai total nul, de ceux qui appartiennent déjà à cette ligne.

Pour tout cvi appartenant à la Ième ligne,
faire pour tout cvj appartenant à DES(cvi),
si le délai total du chemin joignant cvi à cvj est nul
alors cvj appartiendra à la ligne I.

On remarque qu'au départ de la récurrence, on peut placer le cvd en D(0,1).

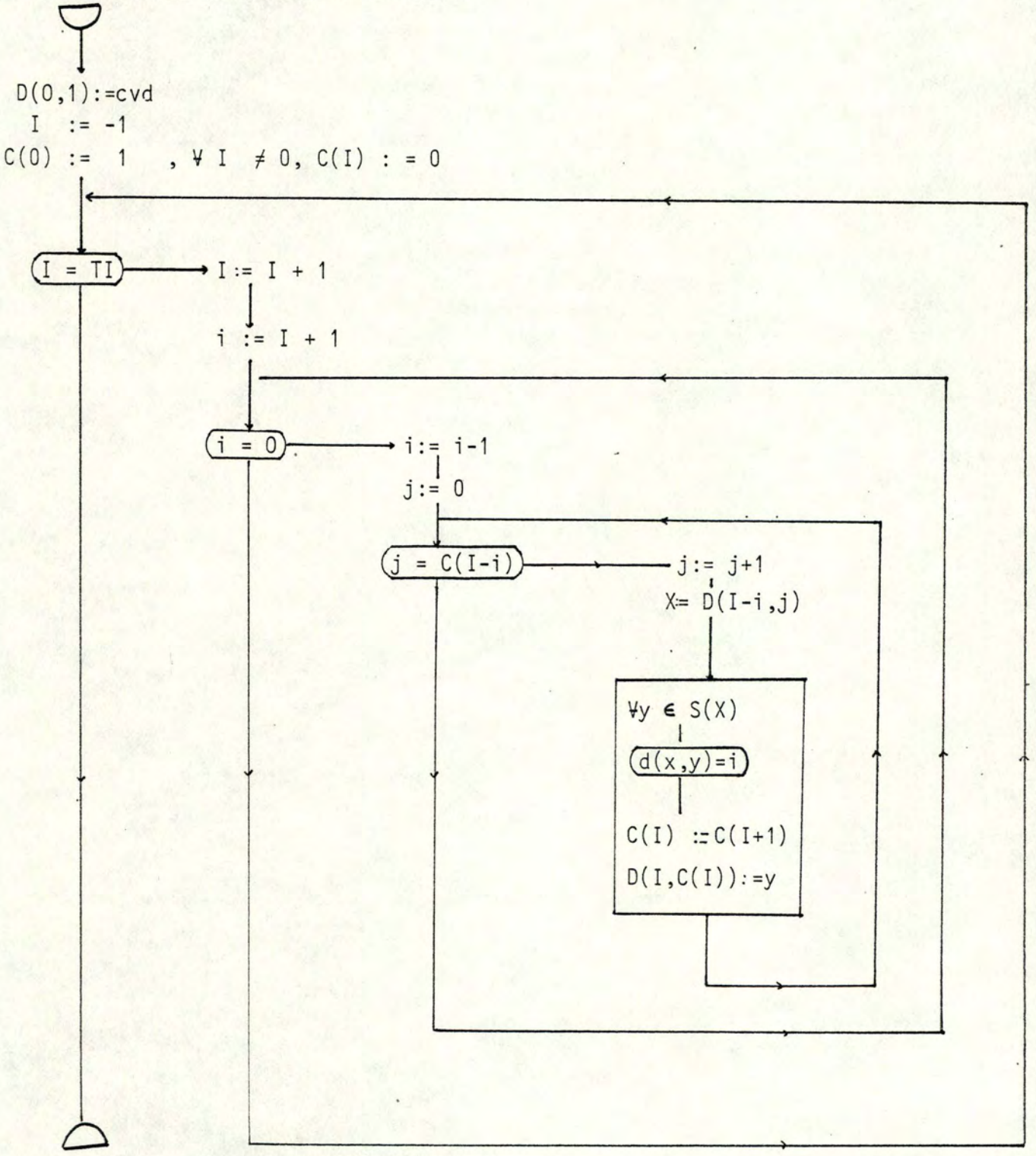
De plus, on constitue un vecteur C(0:TI) tel que C(I) représente le nombre d'éléments déjà placés sur la ligne I.
Au départ donc, seul C(0) vaut 1 puisqu'on a placé le cvd sur cette ligne 0.
Par la suite, pour placer un nouvel élément sur une ligne I quelconque, il conviendra d'abord d'incrémenter C(I) puis de placer le nouvel élément en D(I, C(I)).

2. Présentation de l'algorithme.

L'algorithme est constitué de quatre boucles imbriquées l'une dans l'autre.
La première boucle s'effectue sur les valeurs de I, allant de 0 à TI (temps imparti).
La seconde boucle s'exécute sur i, indiçant la récurrence.
C'est-à-dire, elle parcourt les lignes de la matrice précédentes à la ligne I. i prendra donc des valeurs entre 0 et I.
A la troisième boucle, j permet de passer en revue tous les éléments d'une ligne I - i et prendra donc des valeurs allant de 1 à C (I - i).
A la quatrième boucle, on parcourt les éléments de S(X) où X est un élément de la ligne I - i. La façon dont cette boucle sera organisée dépendra de l'agencement des données dans le fichier.

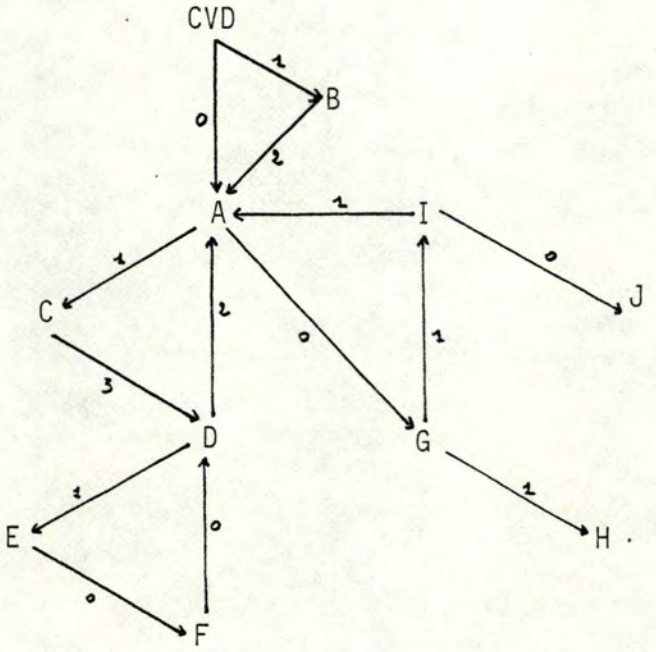
- Algorithme de récurrence -

Par convention, lors d'un test représenté par ce symbole ( ), on considère que le test est négatif si on part sur un des côtés, et positif si on part verticalement.



3. Exemple.

Soit le schéma économique suivant, et soit le temps imparti, TI, valant 6.



L'exécution de l'algorithme provoque les événements suivants :

1°) on place le cvd en D (0,1), I = -1

2°) I ≠ TI = 6

on entre dans la 1ère boucle I = 0

i = 1 ≠ 0

3°) on entre dans la 2ème boucle : i = 0

j = 0 ≠ C(I-i) = C(0) = 1

4°) on entre dans la 3ème boucle

j = 1, X = D (0,1) = CVD

5°) on entre dans la 4ème boucle, quels sont les suivants du CVD de par un arc de délai nul ?

- A

C(0) : = C(0) + 1 = 1 + 1 = 2

et C(0,2) : = A

et on sort de la 4ème boucle.

6°) $j = 1 \neq C(0) = 2$

on rentre dans la 4ème boucle

$j = 2$, $X = D(0,2) = A$

on entre dans la 4ème boucle ⇒ quels sont les suivants de A par un arc de délai nul ?

- G

$C(0) := C(0) + 1 = 2 + 1 = 3$

$D(0,3) := G$

on sort de la 4ème boucle

7°) $j = 2 \neq C(0) = 3$

on sort de la 3ème boucle pour entrer dans la 4ème boucle

$j = 3$, $X = D(0,3) = G$

on entre dans la 4ème boucle ⇒ quels sont les suivants de G par un arc de délai nul ?

- Il n'en existe pas.

on sort de la 4ème boucle.

8°) $j = 3 = C(0) = 3$

on sort de la 3ème boucle

9°) i = 0 vu le 3°) et donc on sort de la 2ème boucle

10°) $I = 0 \neq TI = 6$

on recommence la première boucle, avec la matrice D :

$$D = \begin{pmatrix} CVD & A & G \\ & & \end{pmatrix}$$

On recherche ensuite à établir la liste des sommets influencés après un délai total = 1.

C'est le but de la 1ère boucle, qui s'effectue avec I = 1.

On part des éléments appartenant à D_0 - c'est l'exécution de la 2ème boucle avec i = I = 1,

puisqu'on examine la ligne I - i = 0.

Pour chacun des éléments de la ligne 0, on regarde quels sont les suivants de ceux-ci par un arc de délai égal à 1

- quels sont les suivants du CVD par un arc de délai = 1 : B
- quels sont les suivants de A par un arc de délai = 1 : C
- quels sont les suivants de G par un arc de délai = 1 : H,I

Ceci étant terminé, on sort de la 3ème boucle avec i = 1 ≠ 0.
On recommence cette boucle, i vaut 0.
On va donc s'intéresser aux éléments de la 1ère ligne
((I-i) = (1-0) = 1.)
et voir pour chaque élément quels sont leurs suivants par un arc de délai nul.

- quels sont les suivants de B par un arc de délai = 0 : -
- " " " " de C " " " = 0 : -
- " " " " de H " " " = 0 : -
- " " " " de I " " " = 0 : J
- " " " " de J " " " = 0 : -

On sort de la 4ème et 3ème boucles, i vaut toujours 0, on sort donc également de la 2ème boucle.
On recommence la 1ère boucle avec D maintenant sous la forme suivante :

$$D = \begin{pmatrix} CVD & A & G & & \\ B & C & H & I & J \end{pmatrix}$$

I vaut 2, on s'intéresse donc aux sommets influencés après un délai total valant 2.
i = 2, on regarde les éléments de la ligne I - i; c'est-à-dire la ligne 0.

Parmi les suivants de D_0, quels sont ceux qui le sont par un arc de délai = 2.

- quels sont les suivants de CVD par un arc de délai 2 = -
- " " " " de A " " " 2 = -
- " " " " de G " " " 2 = -

—

i vaut 1, on regarde les éléments de la ligne 1 ,

- quels sont les suivants de B par un arc de délai 1 = -
- " " " " de C " " " 1 = -
- " " " " de H " " " 1 = -
- " " " " de I " " " 1 = A
- " " " " de J " " " 1 = -

i vaut 0, on observe les éléments de la ligne 2

- quels sont les suivants de A par un arc de délai 0 = G
- " " " " de G " " " 0 = -

i vaut 0 et la 2ème boucle se termine, on recommence la 1ère boucle avec

$$D = \begin{pmatrix} CVD & A & G & & \\ B & C & H & I & J \\ A & G & & & \end{pmatrix}$$

On se préoccupe maintenant de la construction de la 3ème ligne, I étant. égal à 3.

Au départ, i vaut 3, donc on observe les éléments de la ligne D_0.

- quels sont les suivants de CVD par un arc de délai 3 = -
- " " " " de A " " " 3 = -
- " " " " de G " " " 3 = -

On sort de la 3ème boucle et on recommence la 2ème avec i = 1.
Et on prête attention aux éléments de D_1.

- quels sont les suivants de B par un arc de délai 2 = A
- " " " " de C " " " 2 = -
- " " " " de H " " " 2 = -
- " " " " de I " " " 2 = -
- " " " " de J " " " 2 = -

Ensuite, on examine les suivants d'éléments de D_2

- quels sont les suivants de A par un arc de délai 1 = C
- " " " " de G " " " 1 = H,I

Enfin, on observe les sommets qui viennent d'être placés en D_3

- quels sont les suivants de A par un arc de délai 0 = G
- " " " " de C " " " 0 = -
- " " " " de H " " " 0 = -
- " " " " de I " " " 0 = J
- " " " " de G " " " 0 = -
- " " " " de J " " " 0 = -

D devient

CVD	A	G			
A	C	H	I	J	
A	G				
A	C	H	I	G	J

Quels sont maintenant les sommets de délai total = 4

D_4 = (D C A H I G J)

puis D_5 = (E C A H I F G J D)

et D_6 = (D C A H I E G J F D)

4. Preuve théorique de la correction de l'algorithme.

On utilisera,pour démontrer la correction de l'algorithme, la méthode de l'invariant qui sera tout d'abord exposée.

On dira qu'un programme est correct s'il respecte sa spécification, c'est-à-dire s'il fait bien ce qu'il a à faire.

La méthode de l'invariant est composée de trois étapes :

1°) On montre qu'à chaque entrée dans la boucle, et pour des valeurs initiales données, les variables vérifient une certaine propriété I, appelée de ce fait l'invariant.

2°) On démontre que si le programme se termine, il aura bien effectué ce qu'il fallait, il aura respecté ses spécifications.

3°) Il reste à démontrer que le programme se termine.

On démontrera la correction de l'algorithme de récurrence en montrant la correction de chacune des boucles qui le composent.

Spécification du programme

Par un graphe donné, représentant un schéma économique, pour une valeur initiale TI' de TI, et pour un cvd choisi arbitrairement, l'exécution du programme fournit une liste des sommets du graphe, extrémités de cvd-chemins, différents, de délai total inférieur ou égal à TI'.

En d'autres termes, l'exécution du programme fournit l'ensemble des cv influencés par une variation du cvd dans un délai inférieur ou égal au temps imparti. De plus, un cv sera représenté, dans cet ensemble, autant de fois qu'il est extrémité de cvd-chemins différents,de délai total inférieur ou égal à TI'.

Correction de la 4ème boucle.

Soit X' un sommet du graphe

I' un entier tel que $0 \leqslant I' \leqslant TI$

i' un entier quelconque

C' un entier $\geqslant 0$

nb' le nombre d'arcs de délai i' issus de X'.

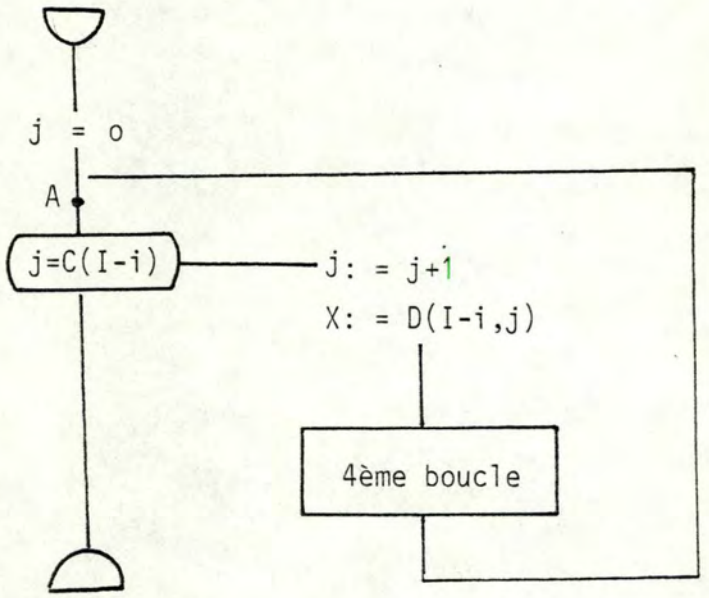
L'exécution de la boucle avec des valeurs initiales

i = i' , I = I' , c(I') = C' et X = X'

- établit C(I') = C' + nb'
- place sur D (I' , C' + 1), ..., D (I', C' + nb') les sommets égaux aux extrémités des arcs, de délai i', issus de X'
- ne modifie pas les valeurs de i et I, ni les valeurs des autres éléments de C et D.

On supposera cette boucle correcte pour le moment, puisqu'elle n'est pas encore définie clairement. Il faut, pour créer cette boucle de l'algorithme,connaître la structure des données, qui sera exposée ultérieurement.

Correction de la 3ème boucle.



Spécification :

Soit I' entier tel que $0 \leqslant I' \leqslant TI$

i' entier tel que $0 < i' \leqslant I'$

m' , n' entiers $\geqslant 0$

$X_1, \ldots, X_{n'}$ suite de sommets de graphe

$narc' = \sum_{\mu=1}^{n'}$ nombre d'arcs de délai i' issus de X_μ

$nchem = \sum_{\mu=1}^{n'}$ nombre de chemins différents, de délai total nul, issus de X_μ

L'exécution de la boucle avec $\underline{i = i' > 0}$, I = I' ,
C(I' - i') = n' , C(I') = m',
et (D(I' - i' , 1)),...., D(I' - i' , n') = $(X_1, \ldots, X_{n'})$

- établit C(I') = m' + narc'
- place sur D(I' , m' + 1), ..., D(I', m' + narc') les extrémités des arcs issus de $X_1, \ldots, X_{n'}$ de délai i'
- ne modifie pas les valeurs de i et I, ni des autres éléments de C et D.

L'exécution de la boucle avec $\underline{i' = 0,}$ I = I' , c(I') = m' et
(D(I', 1), ..., D(I', m')) = $(X_1, \ldots, X_{m'})$

- établit C(I') = m' + nchem
- place sur D(I', m' + 1), ..., D(I', m' + nchem) les extrémités des chemins différents, de délai total nul issus de $X_1, \ldots, X_{m'}$
- ne modifie pas les valeurs de I,i ni les autres éléments de C et D.

Démonstration

On démontrera séparément les cas où i = 0 et i ≠ 0.

Cas i ≠ 0

1ère étape de la méthode de l'invariant.

On montre qu'à chaque passage au point A, les variables vérifient I_A

I_A : 1) $0 \leqslant j \leqslant n'$

2) soit $X_1, \ldots, X_j$ les j premiers sommets de la suite $X_1, \ldots X_{n'}$

et soit narcj = $\sum_{\mu=1}^{j}$ nombre d'arcs de délai i' issus de X_μ

on a C(I') = m' + narcj

D(I', m' + 1), ..., D(I', m' + narcj) représentent les extrémités des arcs issus de $X_1, \ldots, X_j$ de délai i'.

On démontre cela par récurrence

- lors du premier passage
  1) $0 \leqslant j \leqslant n'$ puisque j = 0
  2) $X_1, \ldots, X_0 = \emptyset \iff$ narcj = 0 et C(I') = m'
     D n'a pas changé. La deuxième proposition est donc vérifiée.
- Si I_A est vérifié lors du nème passage, on montre qu'il l'est également au passage suivant.

I_A est vérifié lors du nème passage

1) $0 \leqslant j_n \leqslant n'$
   et même $j_n < n'$ sinon il ne pourrait pas y avoir de passage suivant.

2) soit $X_1, \ldots, X_{jn}$ les j_n premiers éléments de la suite $X_1, \ldots, X_{n'}$
   et narcj_n = $\sum_{\mu=1}^{j_n}$ nombre d'arcs de délai i' issus de X_μ

on a $C(I') = m' + narcj_n$

et $D(I', m' + 1), \ldots\ldots, D(I' + m' + narcj_n)$ sont les extrémités des arcs de délai i', issus de $X_1, \ldots\ldots X_{jn}$.

Que se passe-t-il au n + 1ème passage ?

1) $j_{n+1} = j_n + 1$ et comme $0 \leqslant j_n < n'$
on a $0 \leqslant j_{n+1} = j_n + 1 \leqslant n'$

2) l'exécution de la 3ème boucle comprend l'exécution de la 4ème boucle,
avec X_{jn+1} comme valeur de X'
et m' + narcjn comme valeur de C'

les spécifications de la 4ème boucle établissent

- $C(I') = m' + narcj_n$ + nombre d'arcs de délai i' issus de X_{jn+1}
$= m' + narcj_{n+1}$,

- $D(I', m' + narcj_n + 1), \ldots\ldots, D(I', m' + narcj_{n+1})$ sont les sommets extrémités des arcs de délai i', issus de $X_{j_{n+1}}$
et donc $D(I', m' + 1), \ldots\ldots D(I', m' + narcj_{n+1})$ sont les extrémités des arcs de délai i', issus de
$X_1, \ldots\ldots, X_{j_{n+1}}$

et la deuxième proposition de l'invariant est démontrée.

2ème étape de la méthode de l'invariant.

Il faut montrer que si la boucle se termine, elle établit ses spécifications.

Or la boucle se terminera pour une valeur de j égale à n' = C(I'-i') et lors du dernier passage en A, on a I_A vérifié, dès lors soit X_1, ..., $X_{n'}$ la suite des sommets

et narc' = $\sum_{\mu=1}^{n'}$ nombre d'arcs de délai i' issus de X_μ

On a C(I') = m' + narc'

et D(I', m' + 1), ..., D(I', m' + narc') représentent les sommets extrémités des arcs issus de X_1, ..., $X_{n'}$ de délai i',

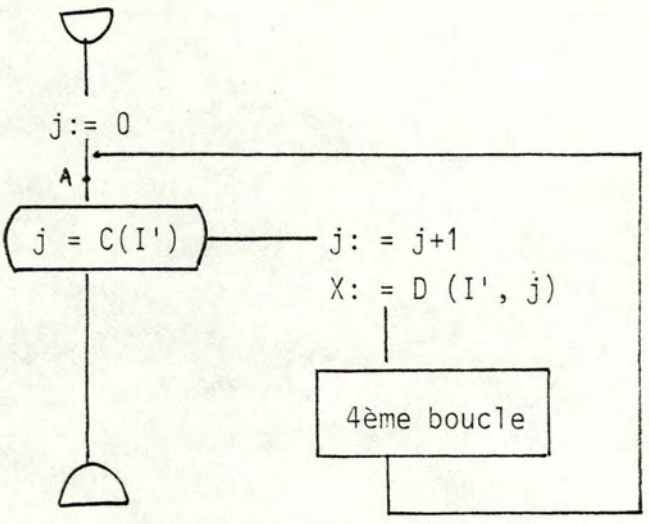
les valeurs de i et I, ainsi que les autres éléments de C et D n'ayant pas été modifiés, on a bien ce que l'on désirait.

3ème étape de la méthode de l'invariant.

On montre que la boucle se termine.
C'est évident puisuqe j s'incrémente d'une unité à chaque entrée dans la boucle, et donc j atteint la valeur n'= C(I'-i') finie après un nombre fini de tours de boucle.

Cas où i' = 0

La boucle devient



1°) Soit $\mathcal{X}$ un ensemble de sommets de graphe,
on définit max ($\mathcal{X}$) comme étant la longueur maximale atteinte par le plus long chemin de délai total nul, issu d'un quelconque des sommets de $\mathcal{X}$

$$\max(\mathcal{X}) = \max_{l} \{ l = \text{longueur des chemins de délai total nul issus des sommets de } \mathcal{X} \}.$$

On remarque que s'il existe NS sommets dans un graphe G ($\mathcal{X}$,U) représentant un schéma économique, on pourra dire que max ($\mathcal{X}$) $\leqslant$ NS-1. Sinon,il y aurait un circuit de délai total nul, ce qui est exclu dans ce contexte.

2°) Soit n'_λ $(X_1, \ldots\ldots, X_{m'})$ = le nombre de chemins de longueur λ, de délai total nul, issus de $X_1, \ldots\ldots, X_{m'}$ avec λ prenant des valeurs entre 0 et max $(X_1, \ldots\ldots, X_{m'})$
avec $n'_0 = m'$ par convention
soit nchem_λ $(X_1, \ldots\ldots, X_{m'}) = \sum_{k=0}^{\lambda} n'_k(X_1, \ldots\ldots, X_{m'})$.

On montre que $\text{nchem}_{\max}$ $(X_1, \ldots\ldots, X_{m'})$ est fini

car le nombre de sommets est fini = m'
la longueur max est finie, et est de NS-1 = m'-1
puisqu'il n'y a pas de circuit de délai total nul.
Donc le nombre de chemins de longueur inférieure ou égale à max, de délai total nul, issus de $(X_1, \ldots\ldots, X_{m'})$ est fini.

1ère étape de la méthode de l'invariant.

A chaque passage au point A, les variables vérifient la propriété suivante :

I_A : 1°) $0 \leqslant j \leqslant \text{nchem}_{\max}$

2°) $\exists\, \lambda : 0 \leqslant \lambda \leqslant \max$
tel que

a.- $D(I', 1), \ldots, D(I', nchem_{\lambda-1})$ représentent les sommets extrémités de chemin de délai total I', issus du cvd, se terminant par au plus $\lambda - 1$ arcs de délai nul.

Par convention, $n'_{-1} = 0$ et $nchem_{-1} = 0$

b.- $nchem_{\lambda-1} + 1 \leqslant j \leqslant nchem_{\lambda}$, $\lambda > 0$
et $0 \leqslant j \leqslant nchem_{\lambda}$ pour $\lambda = 0$

c.- $D(I', nchem_{\lambda-1} + 1), \ldots, D(I', nchem_{\lambda})$ représentent les sommets extrémités de cvd - chemins de délai total I', se terminant par exactement λ arcs de délai nul;

d.- $nchem_{\lambda} \leqslant C(I') \leqslant nchem_{\lambda+1}$

e.- $D(I', nchem_{\lambda}+1), \ldots, D(I', C(I'))$ représentent les sommets extrémités d'un arc de délai nul issu d'un sommet parmi $D(I', nchem_{\lambda-1}+1), \ldots, D(I', j)$

Voir illustration page 28'

Démonstration

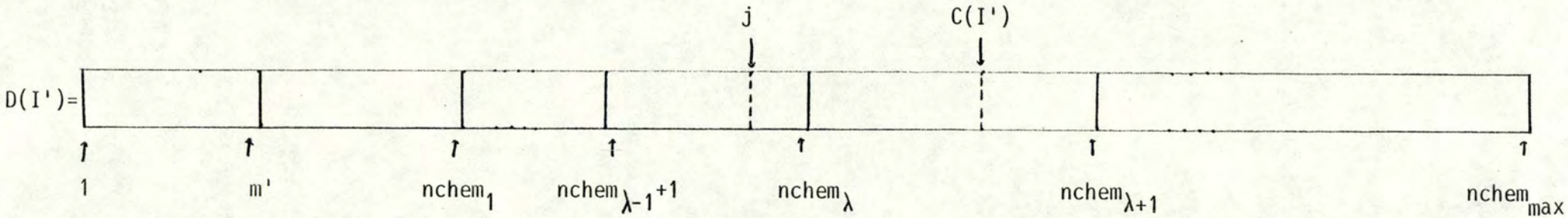
- Lors du 1er passage en A, $j = 0$ et $C(I') = m'$
on a donc bien :

1) $0 \leqslant j \leqslant nchem_{max}$

2) on prend $\lambda = 0$

a. $D(I', 1), \ldots, d(I',0) = \emptyset$ et représente bien les sommets extrémités de cvd-chemin de délai I', se terminant par au plus -1 arc de délai nul.

b. $\lambda = 0$
$0 \leqslant j \leqslant m'$ puisque $j = 0$



Sommets extrémités de cvd-chemins de délai I', ne se terminant pas par un arc de délai nul

Sommets extrémités de cvd-chemins de délai I' se terminant par exactement λ arcs de délai nul

Sommets extrémités de cvd-chemins de délai I', se terminant par un et un seul arc de délai nul

Sommets extrémités d'arcs de délai nul issus de sommets appartenant à $D(I', nchem_{\lambda-1}+1), \ldots, D(I',j)$

c. D(I', 1), ..., D(I', m') représentent les sommets extrémités des cvd-chemins de délai I', se terminant par 0 arc de délai nul. Ce sont les sommets qui ont été placés, par cette 3ème boucle, pour des valeurs de i allant de I' à 1.

d. $m' \leqslant C(I') = m'$

e. d(I', m' + 1), ..., D(I', C(I')) = $\emptyset$ et représente bien les sommets extrémités d'un arc de délai nul issu d'un sommet parmi
D(I', 1), ..., D(I', j)
c'est-à-dire D(I', 1), ...,D(I', 0) = $\emptyset$

- Lors du nème passage, on a

1) $0 \leqslant j \leqslant nchem_{max}$

et même $0 \leqslant j_n < nchem_{max}$

car si $j_n = nchem_{max}$

la 2e proposition de l'invariant sera vérifiée pour λ = max

or d $\longrightarrow$ $nchem_{max} \leqslant C_n(I') \leqslant nchem_{max} + 1$

et,par définition de max, $nchem_{max} = nchem_{max} + 1$

donc $C_n(I') = nchem_{max} = j_n$

et il n'y aurait pas de passage suivant possible.

2) $\exists\ \lambda_n$: $0 \leqslant \lambda_n < max$
tel que

a. D(I', 1), ..., D(I', $nchem_{\lambda_n - 1}$) représentent les sommets extrémités de cvd-chemins de délai I' se terminant par au plus $\lambda_n - 1$ arcs de délai nul.

b. $nchem_{\lambda_n - 1} + 1 \leqslant j_n \leqslant nchem_{\lambda_n}$

c. D(I', $nchem_{\lambda_{n-1}} + 1$), ..., D(I', $nchem_{\lambda_n}$) représentent les sommets extrémités de cvd-chemins de délai total I' se terminant par exactement λ_n arcs de délai nul.

d. $nchem_{\lambda_n} \leqslant C_n(I') \leqslant nchem_{\lambda_{n+1}}$

e. D(I', $nchem_{\lambda_n} + 1$), ..., D(I', $C_n(I')$) représentent les sommets extrémités d'un arc de délai nul issu d'un sommet parmi
D(I', $nchem_{\lambda_{n-1}} + 1$), ..., (D(I', j).

Lors du passage suivant, on a : $j_{n+1} = j_n + 1$

et la 4ème boucle s'est exécutée avec

$X' = D(I', j_{n+1})$

$C' = C_n(I')$

on a

1) $0 \leqslant j_n \leqslant nchem_{max}$

car $0 \leqslant j_n < nchem_{max}$

$\Rightarrow 0 \leqslant j_{n+1} = j_n + 1 \leqslant nchem_{max}$

2) l'exécution de la 4ème boucle assure

- $C_{n+1}(I') = C_n(I')$ + nombre d'arcs de délai nul issus de X'
- $D(I', C_n(I'))$, ..., $D(I', C_{n+1}(I'))$ = extrémité des arcs de délai nul issus de X'.

On prend $\lambda' = \lambda_{n+1} = \lambda_n + 1$ si $j = nchem_{\lambda_n}$

et $\lambda' = \lambda_{n+1} = \lambda_n$ si non

a. $D(I', 1)$, ..., $D(I', nchem_{\lambda'-1})$ représentent-ils les sommets extrémités des cvd-chemins de délai total I' se terminant par au plus $\lambda'-1$ arcs de délai nul ?

Si $\lambda' = \lambda_n$ alors ceci est évidemment vrai puisque I_A est vérifié pour $j = j_n$

si $\lambda' = \lambda_n + 1$, il faut montrer que
$D(I', 1)$, ..., $D(I', nchem_{\lambda_n})$ représentent les extrémités des cvd-chemins de délai total I' se terminant par au plus λ_n arcs de délai nul
or, $\lambda' = \lambda_{n+1} \Longleftrightarrow j_n = nchem_{\lambda_n}$
et par c de I_A avec $j = j_n$, on a

$D(I', nchem_{\lambda_{n-1}} + 1)$, ..., $D(I', nchem_{\lambda_n})$ représentent les sommets extrémités de cvd-chemins de délai total I', se terminant par exactement λ_n arcs de délai nul.

Et puisque a. est également vérifié pour $\lambda = \lambda_n$,
on a bien ce qu'il fallait démontrer.

b. $\text{nchem}_{\lambda'-1} + 1 \leqslant j_{n+1} \leqslant \text{nchem}_{\lambda'}$

si $\lambda' = \lambda_n$, $\text{nchem}_{\lambda'-1} + 1 \leqslant j_n + 1 \leqslant \text{nchem}_{\lambda_n}$

car $j_n \neq \text{nchem}_{\lambda_n}$

et par b. de I_A avec $j = j_n$, on avait $\text{nchem}_{\lambda_{n-1}} + 1 \leqslant j_n \leqslant \text{nchem}_{\lambda_n}$

si $\lambda' = \lambda_n + 1$, c'est-à-dire si $j_n = \text{nchem}_{\lambda_n}$

$\text{nchem}_{\lambda_n} + 1 \leqslant j_{n+1} \leqslant \text{nchem}_{\lambda_{n+1}}$ et $\lambda_n \neq \max$

c. $D(I', \text{nchem}_{\lambda'-1} + 1), \ldots, D(I', \text{nchem}_{\lambda'})$ représentent les sommets extrémités de cvd-chemins de délai total I', se terminant par exactement λ' arcs de délai nul.
Si $\lambda' = \lambda_n$, ceci équivaut au c précédent qui est vérifié par hypothèse récurrence.
Si $\lambda' = \lambda_n + 1 \iff j_n = \text{nchem}_{\lambda_n}$
$D(I', \text{nchem}_{\lambda_n} + 1), \ldots, D(I', \text{nchem}_{\lambda_{n+1}})$ représentent les sommets extrémités de cvd-chemins de délai total I', se terminant par exactement $\lambda_n + 1$ arcs de délai nul

car si $j_n = \text{nchem}_{\lambda_n}$

alors $C_n(I') = \text{nchem}_{\lambda_{n+1}}$ puisque par e. on a

$D(I', \text{nchem}_{\lambda_n} + 1), \ldots, D(I', C_n(I'))$ représentent les sommets extrémités d'un arc de délai nul issu d'un sommet parmi
$D(I', \text{nchem}_{\lambda_{n-1}} + 1), \ldots, D(I', \text{nchem}_{\lambda_n})$
par définition de $\text{nchem}_{\lambda_{n+1}}$, $C_n(I') = \text{nchem}_{\lambda_{n+1}}$

et ceci ajouté au c de I(A) avec $j = j_n$ donne bien la thèse.

d. $\text{nchem}_{\lambda'} \leqslant C_{n+1}(I') \leqslant \text{nchem}_{\lambda'+1}$

si $\lambda' = \lambda_n$, c'est-à-dire $j_n \neq \text{nchem}_{\lambda_n}$

par d. on avait $\text{nchem}_{\lambda_n} \leqslant C_n(I') \leqslant \text{nchem}_{\lambda_{n+1}}$

or $C_{n+1}(I') = C_n(I')$ + nombre d'arcs de délai nul issus de $D(I', J_{n+1})$

donc $\text{nchem}_{\lambda_n} \leqslant C_{n+1}(I')$

d'autre part $\text{nchem}_{\lambda_{n+1}} = \text{nchem}_{\lambda_n} + \sum_{i=\text{nchem}_{\lambda_{n-1}}}^{\text{nchem}_{\lambda_n}} n_0'(D(I',i))$

$$\geqslant \text{nchem}\ n + \sum_{i=\text{nchem}_{\lambda_{n-1}}}^{j_n+1} n_0'(D(I',i))$$

puisque $j_{n+1} = j_n + 1 \leqslant \text{nchem}_{\lambda_n}$

$$\geqslant C_{n+1}(I')$$

si $\lambda' = \lambda_n + 1$, c'est-à-dire $j_n = \text{nchem}_{\lambda_n}$

a-t-on $\text{nchem}_{\lambda_{n+1}} \leqslant C_{n+1}(I') \leqslant \text{nchem}_{\lambda_{n+2}}$?

or quand $j_n = \text{nchem}_{\lambda_n}$, alors $C_n(I') = \text{nchem}_{\lambda_{n+1}}$

et $C_{n+1}(I') = C_n(I')$ + nombre d'arcs de délai nul issus de $D(I', j_{n+1})$

$= C_n(I')$ + nombre d'arcs de délai nul issus de $D(I', \text{nchem}_{\lambda_n} + 1)$

donc $C_{n+1}(I') \geqslant \text{nchem}_{\lambda_{n+1}}$

d'autre part $\text{nchem}_{\lambda_{n+2}} = \text{nchem}_{\lambda_{n+1}} + \sum_{i=\text{nchem}_{\lambda_n}}^{\text{nchem}_{\lambda_{n+1}}} n_0'(D(I',i))$

$$\geqslant \text{nchem}_{\lambda_{n+1}} + \sum_{i=\text{nchem}_{\lambda_n}}^{j_n+2} n_0'(D(I_0,i))$$

$$\geqslant C_{n+1}(I')$$

e. $D(I', nchem_{\lambda'+1}), \ldots, D(I', C_{n+1}(I'))$ représentent les sommets extrémités des arcs de délai nul issus d'un sommet parmi
$D(I', nchem_{\lambda'-1} + 1), \ldots, D(I', j_{n+1})$

si $\lambda' = \lambda_n$, c'est-à-dire $j_n \neq nchem_{\lambda_n}$

$D(I', nchem_{\lambda_n} + 1), \ldots, D(I', C_n(I'))$ représentent les sommets extrémités des arcs de délai nul issus d'un sommet parmi
$D(I', nchem_{\lambda_{n-1}} + 1), \ldots, D(I', j_n)$
par e. de I_A avec $j = j_n$

or l'exécution de la 4ème boucle assure
$D(I', C_n(I')), \ldots, D(I', C_{n+1}(I'))$ représentent les extrémités des arcs de délai nul issus de $D(I', j_{n+1})$. En rapprochant ces deux assertions, on obtient la thèse recherchée.

Si $\lambda' = \lambda_n + 1$, c'est-à-dire $j_n = nchem_{\lambda_n}$

on sait que $C_n(I') = nchem_{\lambda_{n+1}}$

Il faut montrer que $D(I', nchem_{\lambda_{n+1}} + 1), \ldots, D(I', C_{n+1}(I'))$ représentent les extrémités des arcs de délai nul issus de sommets parmi

$$\underbrace{D(I', nchem_{\lambda_n} + 1), \ldots, D(I', j_{n+1})}_{D(I', nchem_{\lambda_n} + 1)}$$

et les spécifications de la 4ème boucle assurent cette démonstration.

2ème étape de la méthode de l'invariant.

Il faut montrer que si la boucle se termine, elle établit ses spécifications. Or, la boucle se termine lorsque $j = C(I')$
et, dans ce cas, par b. et d. de I_A, on a $j = C(I') = nchem_{\lambda}$
par e., on a $D(I', nchem_{\lambda} + 1), \ldots, D(I', C(I'))$ représentent les sommets

extrémités des arcs de délai nul issus d'un sommet parmi
$D(I', nchem_{\lambda-1} + 1), \ldots, D(I', j)$
or $D(I', j) = D(I', C(I'))$
Cela veut dire qu'il n'existe plus d'arcs issus de
$D(I', nchem_{\lambda-1} + 1), \ldots, D(I', C(I'))$
et donc puisque $C(I') = nchem_{\lambda}$, λ vaut max
et, par a. et c., on aura :
$D(I', 1), \ldots, D(I', nchem_{max})$ représentent les sommets extrémités de cvd-chemins de délai total I', se terminant par au plus max arcs de délai nul.

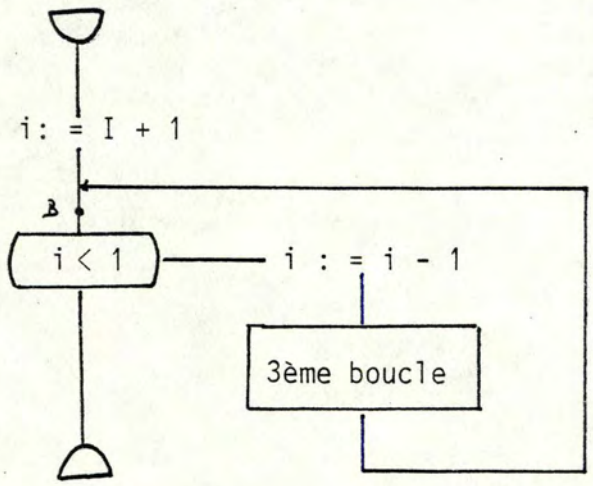
Autrement dit, puisque $D(I', 1), \ldots, D(I', m')$ n'ont pas été modifiés, et vu la méthode de construction de la suite de la ligne I', on a bien placé sur $D(I', m' + 1), \ldots, D(I', nchem_{max})$ les extrémités des chemins de délai total nul, issus de $X_1, \ldots, X_{m'}$.

3ème étape de la méthode de l'invariant.

La boucle se termine car C(I') est au plus égal à $nchem_{max}$ et $nchem_{max}$ est fini pour autant qu'il n'y ait pas de circuit de délai total nul dans le graphe.
D'où la raison de l'hypothèse 2) portée sur les schémas économiques (cf. 1ère partie B .).

j s'incrémentant d'une unité à chaque tour de boucle, il atteint la valeur C(I') après un nombre fini d'entrées dans la boucle.

Correction de la 2ème boucle.



Spécification :

soit E_μ = l'ensemble des sommets du graphe, extrémités de chemins différents de délai total μ issus du cvd

soit I', entier tel que $0 \leqslant I' \leqslant TI$

si, initialement - chacun des éléments C(1), ..., C(I' - 1) a une valeur

- $\{D(\mu, 1), \ldots, D(\mu, C(\mu))\} = E_\mu$ pour $0 \leqslant \mu \leqslant I'$
- I = I'

l'exécution de la boucle

- affecte une valeur $\geqslant 0$ à C(I') = nombre de cvd-chemins différents de délai total I',
- établit $D(I', 1), \ldots, D(I', C(I')) = E_{I'}$,
- ne modifie pas C(1), ..., C(I' - 1), ni les lignes D(1), ..., D(I' - 1), ni la valeur de I.

Démonstration

1ère étape de la méthode de l'invariant

A chaque passage en B, les variables vérifient I_B

I_B : 1) $0 \leqslant i \leqslant I' + 1$

2) si $i \neq 0$

soit $narc_{I-i}$ = nombre d'arcs de délai i issus de $X_1, \ldots, X_{C(I-i)}$, les éléments de la I - ième ligne

on a $C(I') = \sum_{v=i}^{I} narc_{I-v}$

et $D(I', 1), \ldots, D(I', C(I'))$ sont les extrémités des arcs de délai μ issus des éléments de la I - μème ligne pour μ allant de I à i.

si $i = 0$

soit $nchem_{I'}$ = nombre de chemins de délai total nul issus de $X_1, \ldots, X_{C(I')}$, les éléments de la I'ème ligne

on a $C(I') = \sum_{v=1}^{I} narc_{I-v} + nchem_{I'}$

et $D(I', 1), \ldots, D(I', C(I'))$ = extrémités des cvd-chemins de délai total I'.

On démontre par I_B par récurrence

- lors du 1er passage, $i = I' + 1$,
  et donc 1) $0 \leqslant i \leqslant I' + 1$
  2) $i \neq 0$ car $I' \geqslant 0$
  la 2ème proposition est évidemment vraie

- lors du nème passage, on a $i = i_n$, et I_B vérifie

1) $0 \leqslant i_n \leqslant I' + 1$ et même $0 < i_n \leqslant I' + 1$, sinon il ne peut y avoir de passage suivant

2) $i_n > 0$

soit $narc_{I-i_n}$ = nombre d'arcs de délai i_n issus de $X_1, \ldots, X_{C(I - i_n)}$

on a $C(I') = \sum_{v = i_n}^{I} narc_{I-v}$

et $D(I', 1), \ldots, D(I', C(I'))$ sont les extrémités des arcs de délai μ issus des éléments de la $I - \mu$ème ligne, pour μ allant de I à i_n

- lors du $n + 1$ ème passage, $i_{n+1} = i_n - 1$

1°) $0 \leqslant i_{n+1} = i_n - 1 \leqslant I' + 1$

puisque $0 < i_n < I' + 1$

2°) si $i_n - 1 > 0$

et $narc_{I-i_{n+1}}$ = nombre d'arcs de délai $i_{n+1} = i_n - 1$ issus de $X_1, \ldots, X_{C(I-i_{n+1})}$, éléments de la $(I-i_n) + 1$ère ligne

$C(I') \stackrel{?}{=} \sum_{v = i_{n+1}}^{I} narc_{I-v}$

et $D(I', m' + 1), \ldots, D(I', m' + narc\ I - i_{n+1})$? sommets égaux aux extrémités des arcs issus de $X_1, \ldots, X_{C(I-i_{n+1})}$, de délai $i_{n+1} = i_n - 1$

puisqu'au nème passage $C(I') = \sum_{v = i_n}^{I} narc_{I-v}$

et $D(I', m' +1), \ldots, D(I', m' + narc\ I - i_n)$
représentent les sommets, extrémités des arcs issus de $X_1, \ldots, X_{C(I - i_n)}$, de délai i_n

il reste à prouver que l'exécution de la 3ème boucle a permis de rajouter $narc_{I-i_{n+1}} = narc_{I-(i_n-1)} = narc_{I-i_{n+1}}$ à $C(I')$, et a permis de placer en $D(I', m' + narc_{I-i_n}), \ldots, D(I', m' + narc_{I-i_{n+1}})$ les sommets extrémités des arcs issus de $X_1, \ldots, X_{C(I-i_{n+1})}$ de délai $i_n - 1$.

Or, les spécifications de la 3ème boucle garantissent ce résultat puisqu'elle s'exécute avec $i' = i_{n+1} = i_n - 1$

$n' = C(I' - i_{n+1})$

$X_1, \ldots, X_{n'}$ = éléments de la $I-i_{n+1}$ ème ligne

$m' = C(I')$.

Si $i_n - 1 = 0$, c'est-à-dire $i_n = 1$

et nchem(I') = nombre de chemins de délai total nul issus de $X_1, \ldots, X_{m'}$, les éléments de la I' ème ligne

au passage précédent $C(I') = \sum_{v=1}^{I} narc_{I-v}$

et $D(I', 1), \ldots, D(I', C(I'))$ sont extrémités de cvd-chemins de délai total I'.

Il faut, en fait, démontrer que l'exécution de la 3ème boucle avec i' = 0 ajoute le nombre nchem(I') à C(I').
Or, cela est vérifié de par les spécifications de cette boucle.
De plus, avant exécution de cette boucle, D(I', 1), ... D(I', C(I')) sont les extrémités d'arcs de délai μ issus des éléments de la I - μème ligne, pour μ allant de I à 1.
Et donc, tous ces sommets sont extrémités de chemins de délai total égal à (I' - μ) + μ = I'.
Il reste donc à insérer les descendants de ces sommets, par un chemin de délai nul, pour obtenir tous les sommets extrémités d'un chemin de délai total I'. Ceci est exécuté par la boucle 3, où i' = 0.

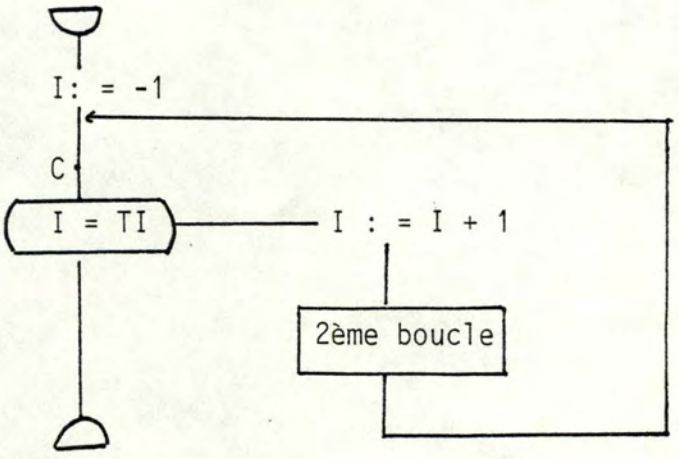
2ème étape de la méthode de l'invariant.

Si la boucle se termine, elle établit ses spécifications, car la boucle se termine pour i < 1, c'est-à-dire pour i = 0, et l'invariant I_B, vérifié quand i = 0, détermine les spécifications de la 2ème boucle.

3ème étape de la méthode de l'invariant.

La boucle se termine car I a une valeur finie, et i se décrémente d'une unité à chaque tour de boucle et atteint finalement la valeur nulle.

Correction de la 1ère boucle.



Spécifications

L'exécution de la boucle - affecte une valeur $\geqslant 0$ à C(0), ..., C(TI), avec TI donné fini

et - établit $D(\mu,1), \ldots, D(\mu, C(\mu)) = E_\mu$

$\forall \mu = 0 \leqslant \mu \leqslant TI$

Démonstration

1ère étape de la méthode de l'invariant.

A chaque entrée dans la boucle, les variables vérifient I_C.

I_C = 1°) $-1 \leqslant I \leqslant TI$

2°) C(μ) = nombre de cvd-chemins différents de délai total μ pour μ de 0 à I

3°) $\{D(\mu, 1), \ldots, D(\mu, C(\mu))\} = E_\mu \quad \forall \mu : 0 \leqslant \mu \leqslant I.$

- lors du 1er passage, I = -1 donc 1) $-1 \leqslant I \leqslant TI$ vérifié
  et, d'autre part, 2) et 3) sont évidemment vérifiés.

- On suppose I_C vérifié pour $I = I_n$

- Lors du n+1ème passage, $I_{n+1} = I_n + 1$

  les spécifications de la 3ème boucle, exécutée avec $I' = I_n + 1$, assurent immédiatement la correction des 2ème et 3ème propositions,
  et la 1ère est également vraie, car lors du nème passage, on avait :

  1) $-1 \leqslant I_n \leqslant TI$ et même $-1 \leqslant I_n < TI$, sinon il ne pouvait y avoir de passage suivant

  et donc $-1 \leqslant I_{n+1} = I_n + 1 \leqslant TI$

2ème étape de la méthode de l'invariant

Si la boucle se termine, I vaudra TI, et lors du dernier passage en C, on aura I_C vérifié et donc on aura les spécifications de la 1ère boucle respectées.

3ème étape de la méthode de l'invariant

La boucle se termine pour autant que le TI donné ait une valeur finie.

5. Complexité théorique de l'algorithme.

Définition:

Lorsqu'on examine un algorithme A, on trouve presque toujours un paramètre, soit p, caractérisant la taille des données auxquelles A s'applique dans chaque cas.

Soit donc T(p) le temps d'exécution de l'algorithme A exprimé en fonction de p.

On dit que l'algorithme A est de complexité théorique O(f(p)) ou f est une certaine fonction du paramètre p si la croissance asymptotique de T(p) est de l'ordre de f(p) ou plus.

Ou, autrement dit, si le rapport T(p)/f(p) est borné, lorsque p tend vers + ∞

Ou encore, s'il existe une constante C telle que T(p) = Cf(p) pour toute valeur positive de p. C'est le cas notamment lorsque T(p)/f(p) tend vers une constante C pour p tendant vers + ∞

Dans le cas de l'algorithme de récurrence présenté précedemment, on constate que le temps d'exécution T dépendra du nombre de flèches du schéma économique (noté NF), ainsi que de la valeur de TI.

T (NF, TI)

La 4ème boucle permet de parcourir les élements suivants d'un sommet X, élément d'une ligne I - i. Pour cela, on parcourt l'ensemble des flèches du graphe pour détecter celle qui ont X pour origine.
La boucle s'effectue donc, au plus, NF fois.

La 3ème boucle s'effectue au plus NC fois où NC est le nombre de colonnes de la matrice. Ce nombre = max $\{$ C(I) : I de 0 à TI $\}$

est difficilement déterminable, on retiendra qu'il est fonction du nombre de flèches.

La deuxième s'effectue au plus TI + 1 fois, car la valeur maximale prise par i est I et I est lui-même maximal quand il vaut TI.

La première boucle ne s'effectuera jamais plus de TI + 1 fois

Finalement, la complexité théorique de l'algorithme de récurrence est de l'ordre de $0 (NF . f(NF) . (TI + 1)^2)$.

6. Calcul de la mesure des influences.

On doit également donner par cet algorithme la mesure des influences qui auront été relevées.
Pour cela, simultanément à la construction de la matrice D, on construit une matrice T, de mêmes dimensions que D, et telle que T(I, J) représente la mesure de l'influence d'une variation de cvd sur le sommet placé en D(I,J).

Cette mesure est le produit des mesures des arcs composant le cvd-chemin menant à D(I,J).

Au départ de la récurrence, on pose T(0,1) = 1, la mesure de l'influence du cvd sur lui-même étant,par convention, l'unité.
Par la suite, dès qu'un élément cvi est placé sur le ligne I,on sait qu'il existe au moins un arc,

- dont l'extrémité soit cvi
- dont l'origine soit un des sommets cvk d'une ligne I-k avec k compris entre 0 et I
- dont le délai soit égal à k.

Dès lors, la mesure de l'influence d'une variation de cvd sur cvi sera le produit de la mesure de l'influence du cvd sur cvk
et de la mesure de l'influence de cvk sur cvi.

$$T(I, C(I)) = T(I-k,j) \times t(\text{arc } (cvk, cvi))$$

avec $D(I, C(I)) = cvi$
$D(I-k, j) = cvk$

DEUXIEME PARTIE

PRÉSENTATION DU PROGRAMME

ET

DES STRUCTURES DE DONNÉES

==========

A. PRESENTATION GENERALE DU PROGRAMME.

Le programme a été réalisé en PASCAL car il doit s'inscrire dans le cadre d'un projet de la Faculté des Sciences Economiques et Sociales qui l'exige.

La spécification du programme est la suivante :

A partir d'un schéma économique donné (éventuellement vide), le programme retourne un schéma économique (éventuellement modifié par rapport au premier) et fournit les résultats de 0, 1 en plusieurs simulations sur des schémas économiques.

Le programme est construit comme une hiérarchie : il est organisé en niveaux distincts et ordonnés.

Au niveau 3 : le corps du programme

Au niveau 2 : on trouve les 3 modules principaux constituant le programme

1) module d'introduction des données
2) module de gestion des diverses modifications possibles sur le schéma donné
3) module d'exécution de l'algorithme présenté en 1ère partie

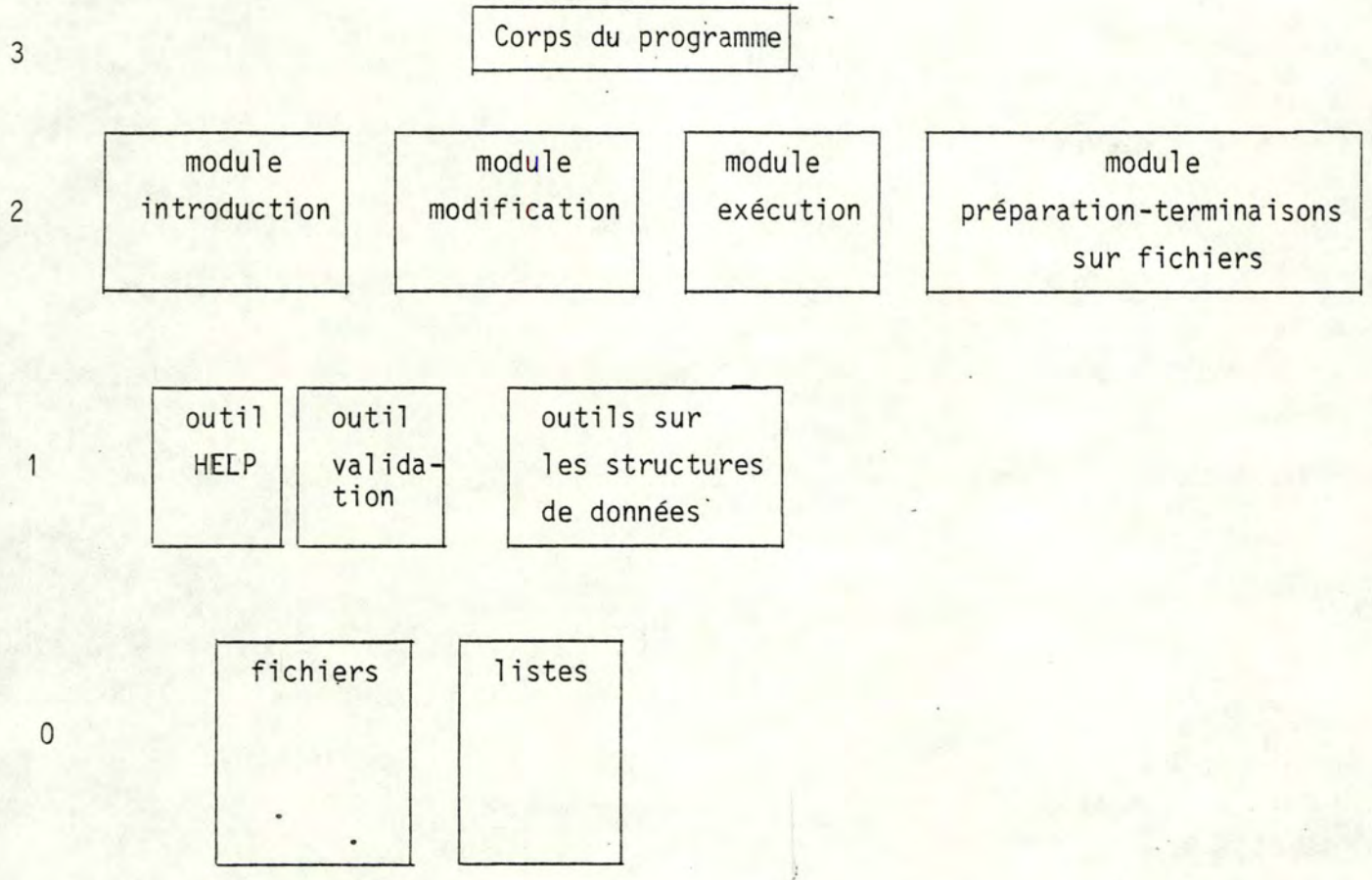
et on y trouve également

4) outils de travail nécessaires à la préparation des fichiers ou à leur sauvetage : module de préparation et terminaison des fichiers.

Au niveau 1 : il existe différents outils :

1) module d'aide à l'utilisateur
2) module de validation des données
3) différents modules régissant les structures de données et l'accès à celles-ci

Au niveau 0 : on trouve les structures de données, c'est-à-dire les fichiers et les listes.



Plus rigoureusement, on dit qu'une structure est hiérarchisée ssi $\exists$ relation R entre les composants qui permet de définir des niveaux tels que

1) le niveau 0 = l'ensemble des composants A
tels que $\nexists$ composant B tel qu'on ait R(A,B)

2) le niveau i = l'ensemble des composants A
tels que - $\exists$B de niveau i-1 tel que R(A,B)
- si R(A,C) alors C est de niveau $\leqslant$ i-1

Dans ce programme, entre le niveau 3 et le niveau 2, on a une relation R = appelle, sauf pour le module des outils de préparation et terminaison sur les fichiers qui lui est "utilisé" par le corps du programme.

Entre les niveaux inférieurs, on a une relation "utilise",
c'est-à-dire R(A,B) ≡ A utilise B

ssi le fonctionnement correct de A dépend de la disponibilité d'une version correcte de B

(correcte ∩ conception ∩ spécification ∩ codage)

Ainsi, pour que le programme respecte sa spécification et puisse retourner un schéma économique, il est nécessaire et suffisant que le module préparation-terminaison des fichiers soit correct.

Par contre, le programme peut se passer du module introduction, si le schéma économique donné en entrée est non vide, par exemple.

Le programme peut également se passer du module de modification et du module d'exécution, car il respectera toujours ses spécifications, s'il renvoie le même schéma économique que celui entré, sans avoir ni modifié, ni exécuté de simulation sur celui-ci.

D'autre part, entre le niveau 2 et le niveau 1, on a une relation utilise.

On a donc bien la hiérarchie annoncée puisque:

1°) le niveau 0 est constitué des composants, données, qui n'utilisent ni n'appellent aucun autre composant

2°) les composants d'un même niveau ne s'appellent ni ne s'utilisent entre eux.

Les avantages d'une hiérarchisation sont nombreux :

- cela permet de restreindre les interrelations entre composants, ce qui facilite la conception des différents composants.

  Si A utilise B, il n'est pas nécessaire de connaître le développement de B pour construire A, seules les spécifications de B sont nécessaires. Et pour concevoir B, on ne doit rien connaître de A.

- la validation est plus facile.
  Les tests se feront niveau par niveau, on ne devra pas nécessairement tout tester d'un coup.

- la maintenance est simplifiée
  car les modifications futures à apporter seront localisées.

B. LE MODULE INTRODUCTION.

Ce module sert à introduire dans un fichier les données qui forment le schéma économique, ceci lorsque le schéma économique donné en entrée est vide, et que le programme le construit.
Ce module correspond à la procédure introduction du programme.

Spécification de la procédure introduction : la procédure constitue interactivement un fichier séquentiel de flèches, tel que les flèches de même origine sont groupées, représentant un schéma économique, ainsi qu'une liste des arcs de délai nul et une liste des sommets du schéma avec leur valeur respective. La procédure renvoie une valeur d'arrêt = 'o' lorsque le schéma est totalement constitué.

Le module est essentiellement composé d'instructions de saisie, de validation des données et de rangement dans les fichiers.
La validation des données porte sur le nom du sommet (en a-t-il un ?, l'extrémité d'une flèche est-elle égale à son origine ?), porte également sur le délai d'un arc (positif et inférieur à limdel, donnée arbitrairement au départ du programme), ainsi que sur le coéfficient de l'arc (compris entre liminfmes et limsupmes, données arbitrairement au départ du programme).

Quelle structure adopter pour ranger les données dans le fichier schéma ?

1. Structure des données.

Pour représenter un graphe, de manière générale, sous forme de fichier, il existe deux solutions :

1°) Créer un fichier dont chaque article représente un sommet.
Pour représenter un schéma économique, on imagine donc un fichier dont chaque article est formé de différents items

a) le nom du sommet ou un identifiant quelconque
cvi par exemple

b) le nombre d'arcs dont le sommet représenté est origine.
Autrement dit, le nombre d'arcs incident à cvi, vers l'extérieur

c) un sous-article qui se reproduit autant de fois que l'indique le nombre recueilli en b).
Chaque sous-article désignera donc une flèche dont le sommet cvi est origine.
Les sous-article comprendront:

1. l'extrémité de l'arc désigné
2. le délai de l'arc désigné
3. le coéfficient, la mesure de l'influence que l'arc représente.

Par exemple, le sommet A de l'exemple 3), partie 1, sera représenté par :

A	2	C	1	t_i
		G	0	t_j

Ceci est parfaitement réalisable en COBOL, grâce à l'instruction "depending on" qui permet de déclarer légalement de tels articles et sous-articles.

Mais, malheureusement, en PASCAL, c'est beaucoup plus difficilement faisable.
Il reste à exploiter la seconde solution.

2°) Créer un fichier où chaque article représente un arc du graphe. Ainsi, pour la représentation d'un schéma économique, chaque article sera composé de 4 items

a) l'origine de l'arc représenté
b) l'extrémité de celui-ci
c) le délai de cet arc
d) la mesure de l'influence qu'il représente.

Si une variation de cvi provoque une variation de cvj, par exemple, après un délai dij et donc une mesure tij, alors l'article qui reflétera cette influence sera le suivant :

cvi	cvj	dij	tij

Ceci est aisément compatible avec le langage PASCAL. Mais pour être effectif, il faut encore établir une organisation des articles du fichier.

Il est important de remarquer que le choix d'une organisation aura des conséquences considérables sur l'élaboration de la quatrième boucle de l'algorithme de récurrence d'une part, et sur la création des procédures destinées à effectuer diverses modifications du schéma économique donné d'autre part.

Ainsi, ajouter un arc entre deux sommets, par exemple, c'est-à-dire insérer un article dans le fichier, se fera différemment selon que le fichier est de structure séquentielle ou séquentielle indexée, par exemple.

2. Organisation des données.

Quelles sont les possibilités d'organisation du fichier ?

A. Organisation séquentielle.

On range les articles les uns à la suite des autres. Cela peut s'opérer sans ordre précis, ce qui complique la recherche des suivants d'un sommet X, envisagée à la quatrième boucle de l'algorithme de récurrence.
Mais on peut également établir un ordre,alphabétique par exemple, sur l'item-origine des articles. Dans ce cas, on aura besoin d'une procédure de tri.
Ainsi, l'exemple 3) de la partie 1 serait représenté par le fichier suivant :

A	C	1	t1
A	G	0	t2
B	A	2	t3
C	D	3	t4
cvd	A	0	t5
cvd	B	1	t6
D	A	2	t7
D	E	1	t8
E	F	0	t9
F	D	0	t10
G	H	1	t11
G	I	1	t12
I	A	1	t13
I	J	0	t14

Dans le programme, plutôt que d'utiliser une procédure de tri, on demande à l'utilisateur d'entrer son schéma économique, origine par origine, et pour chaque origine, arc par arc. Ceci a l'avantage d'orienter l'utilisateur dans son entrée des données, ce qui est utile si le schéma comporte de nombreux arcs.
Mais, d'autre part, si au cours de la procédure d'introduction, l'utilisateur oublie une flèche, il ne pourra la rajouter que par le biais du module de modification du schéma.
Les articles de l'exemple 3), partie 1, seront donc rangés comme suit :

cvd	A	0	t 5
cvd	B	1	t 6
A	C	1	t 2
A	G	0	t 1
B	A	2	t 3
C	D	3	t 4
D	A	2	t 7
D	E	1	t 8
E	F	0	t 9
F	D	0	t10
G	I	1	t12
G	H	1	t11
I	A	1	t13
I	J	0	t14

Ainsi, les arcs de même origine restent-ils groupés. Ce qui facilite la recherche des suivants d'un sommet. La quatrième boucle est donc facilement implémentable sur cette structure.

L'organisation séquentielle a donc l'avantage appréciable d'être simple, donc facile à mettre en oeuvre, et permet une construction aisée de la quatrième boucle de l'algorithme et des procédures de modification du schéma économique.
Cependant, cette organisation est relativement lourde à manipuler, surtout si le fichier comporte de nombreux articles, car le PASCAL offre peu de facilités de gestion de fichiers.

B. Organisation par accès direct.

L'idée est d'associer à chaque sommet un numéro de bloc. Et ce bloc contiendra tous les articles représentant les arcs dont l'origine est ce sommet. Le fichier sera, lui, constitué de la suite des blocs.

Pour réaliser cela, on établit, au moment de l'introduction des données, une table de correspondance entre les sommets et les numéros de bloc.

On devra, de plus, prévoir une taille de bloc (la même pour tous les blocs). On peut l'imaginer maximale, de telle sorte qu'un bloc puisse toujours contenir l'ensemble des articles voulus. Mais cela entraînerait des pertes de place non négligeables.

Cependant, on peut tout aussi bien prévoir une taille moyenne et organiser un système de blocs de débordement, dans lesquels on rangerait les articles qui ne pouvaient être contenus dans le bloc voulu.

Ainsi le schéma économique de l'exemple 3), partie 1, serait-il représenté par la table de correspondance et le fichier suivants :

1. Table de correspondance :

A	1
B	2
C	3
cvd	4
D	5
E	6
F	7
G	8
H	9
I	10
J	11

2. Pour des blocs de la taille de 2 articles, on aura :

bloc 1

C	1	t1
G	0	t2

bloc 2

A	2	t3

bloc 3

D	1	t4

bloc 4

A	0	t5
B	1	t6

et ainsi de suite.

Il est inutile de rappeler l'origine des arcs représentés dans un bloc quelconque, puisqu'on la connaît grâce à la table de correspondance.

On remarque qu'ajouter un arc au schéma et, par là même, insérer un article dans le fichier n'est pas très simple.

Il faudra 1°) accéder au bloc correspondant au sommet origine de l'arc. Au besoin, créer ce bloc, si l'origine n'existe pas encore dans le schéma.

2°) Si le bloc n'est pas rempli, on y ajoute le nouvel article. Sinon, on crée un bloc de débordement dans lequel on rangera cet article.

Mais pour ce faire, il est impératif de rajouter deux renseignements de grande importance à chacun des blocs :

i) le nombre d'articles que le bloc contient, de façon à repérer si oui ou non il y aura encore place dans le bloc pour un nouvel article;

ii) l'adresse ou, plus précisément, le numéro du bloc de débordement dans lequel on range les articles nouveaux qui n'ont pas eu de place dans le bloc initialement prévu.

Néanmoins, si ceci semble bien au point sur papier, ce l'est beaucoup moins en réalité car sur le DEC,en PASCAL, le seul accès direct possible ne se fait pas sur le numéro d'un bloc mais sur l'adresse en mémoire du premier bit d'un bloc.
Cela rend cette organisation plus difficile à implémenter. Mais surtout, cela réduit les perspectives de portabilité. Ce système d'accès direct ne faisait pas partie du PASCAL standard.

C. Autre organisation.

Sur le DEC, en PASCAL, il n'existe aucun autre type d'organisation déjà implémenté. Il faut, si on en veut un nouveau, l'implémenter soi-même. Ce qui repose le problème de la portabilité.
On peut se demander si cela vaut la peine de tels efforts, alors que ce qui est proposé est suffisant et peut même s'avérer franchement valable.

En conclusion, la structure séquentielle est simple et parfaitement implémentable. Elle se prête également sans problème aux manipulations nécessaires à des modifications du schéma économique.
Une telle organisation n'est malheureusement efficace que dans une certaine mesure.
Ainsi, toute modification du fichier entraîne une copie de celui-ci, puisqu'on ne peut écrire les articles dans le fichier que les uns à la suite des autres et qu'il n'est pas possible de se positionner en fin de fichier séquentiel pour l'étendre.
Il existe, en PASCAL non standard, implémenté par DIGITAL, sur le DEC, une instruction APPEND qui permet de se positionner en fin de fichier pour l'étendre, mais ceci n'est valable que pour les fichiers-textes. Cette instruction sera notamment utilisée pour l'impression des résultats des différentes simulations effectuées.
D'autre part, accéder à un article, dans un fichier séquentiel, implique l'accès et la lecture de tous les articles précédents, alors qu'en accès direct, on ne lira au plus que les articles de même origine.

La structure par accès direct a donc l'avantage d'être plus efficace mais, en revanche, elle rend le programme importable et ne se justifierait que pour des schémas comportant un nombre faramineux d'arcs et donc des fichiers gigantesques.
Ici, donc, un choix arbitraire s'impose et on préfèrera la simplicité à moindre efficacité plutôt que l'efficacité et la non-portabilité.

D'autre part, chaque sommet du graphe possède une valeur (qui peut éventuellement être modifiée sous l'influence d'une variation d'un autre sommet). Il faut donc donner des valeurs en entrées, comme faisant partie des données du schéma économique.
On aurait pu les inclure dans les items des articles du fichier séquentiel présenté. Cependant, pour un même schéma, l'utilisateur peut vouloir essayer à la simulation des jeux de valeurs différents.
C'est pourquoi on crée un fichier (fsommet) séquentiel, sans ordre précis de rangement, dont chaque article reprend le nom d'un sommet et sa valeur

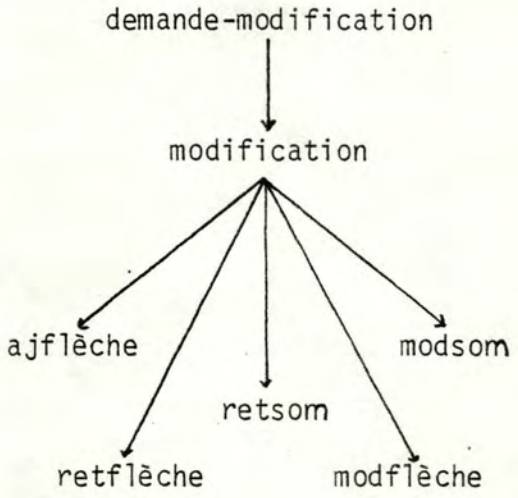
Remarque : pour des raisons de facilité d'exécution du programme (il est plus facile de travailler sur une liste que sur un fichier, notamment pour les modifications), on ne travaille pas sur le fichier (fsommet), mais sur une liste des sommets: la liste somval.
La gestion des opérations de création et suppression sur cette liste, et des accès à celle-ci (accès en avant ou en arrière, accès en début ou fin de liste) est assurée par la procédure listsomval qui est donc un outil de niveau 1.

De même, listarcnul est une liste qui reprend tous les arcs de délai nul qu'on a trouvé dans le schéma donné. Les opérations de création et suppression de cette liste, ainsi que les accès à celle-ci sont gérés par la procédure listarcnul.
Cette liste a été créée pour servir de base à la recherche d'éventuels circuits nuls dans le schéma. Car on a vu que l'algorithme de récurrence ne pouvait pas s'appliquer à des schémas comportant des circuits de délai total nul. Il faut donc vérifier que de tels circuits n'existent pas et listarcnul sera un outil utilisé lors de cette recherche.

C. LE MODULE DE MODIFICATION

La procédure modification dans le programme est exécutée autant de fois que l'utilisateur le désire. C'est pourquoi elle s'inscrit dans la procédure demande-modification.

On a alors la décomposition suivante :



La procédure modification propose à l'utilisateur un menu des diverses modifications possibles.

1) ajflèche : est une procédure qui permet d'ajouter une flèche au schéma, c'est-à-dire qu'elle permet d'inscrire un article nouveau dans le fichier

Spécification :

Pour schéma, un fichier séquentiel de flèches, tel que les articles de même origine sont groupés, la procédure acquiert interactivement un nouvel article et renvoie un fichier dont les articles de même origine sont groupés, identique à schéma, et contenant 0 ou 1 nouvel article (selon que le nouvel article est validé ou pas) ou retourne un message d'erreur si le nouvel article existe déjà, auquel cas schéma est inchangé.
De plus, la procédure met à jour les listes arcnul et somval.

Pour ce faire, on demande l'origine, soit A, et l'extrémité, soit B, du nouvel arc. On valide cet arc comme n'étant pas une boucle.
Si ces sommets n'existent pas encore, on demande leur valeur associée.
Ensuite, puisque le fichier schéma est d'organisation séquentielle, il est impossible d'introduire le nouvel article directement dans ce fichier. Aussi, on parcourt le fichier (schéma) :

- tout article dont l'origine diffère de A est recopié dans un deuxième fichier (schéma 2)
- pour les articles d'origine A, on vérifie que l'extrémité $\neq$ B, sinon cela voudrait dire que la flèche à ajouter existe déjà ! Auquel cas, la procédure se termine par un message d'erreur à l'utilisateur
- lorsqu'on a visité tous les articles d'origine A, on place là le nouvel article dans schéma 2, en ayant, au préalable, demandé et validé les valeurs attachées à cet arc
- si aucun article n'a A pour origine, le nouvel article est placé en fin de fichier.

Cette démarche a été adoptée ici parce qu'elle est relativement aisée à implémenter et surtout parce qu'elle permet de garder groupées les flèches de même origine. Ce qui est approprié à l'exécution de la quatrième boucle de l'algorithme de récurrence.

Les outils du niveau 1 utilisés ici sont :

- listarcnul : car si la nouvelle flèche est de délai nul, il faut l'ajouter à la liste des arcs nuls, ce qui se fait par la procédure de gestion de cette liste : listarcnul.
- listsomval : qui gère la liste dans laquelle on range un nouveau sommet et la valeur qui lui est associée

-validsommet, validtemps, validmesure: valident les données d'un arc.

Spécification de validsommet (n)

Si n vaut 1 : soit début,un sommet
la procédure renvoie une valeur de correct = 'n'
et un message d'erreur approprié si début n'a pas de nom.

Si n vaut 2 : soit fin et début, deux sommets,
la procédure renvoie une valeur de correct = 'n' et
un message d'erreur approprié si fin n'a pas de nom
ou si fin = début.

Si n vaut 3 : soit testé,un sommet
soit message,un caractère
soit somval une liste de sommets
la procédure renvoie une valeur de correct = 'n'
si testé ne se trouve pas dans la liste somval
ainsi qu'un message d'erreur adéquat à condition que
la valeur du message soit 'o'.

Spécification de validtemps

Soit time,limdel, deux entiers,
la procédure renvoie une valeur de correct= 'n'
et un message d'erreur approprié si time est négatif
ou si time est supérieur à limdel.

Spécification de validmesure

Soit coeff. un réel
Soit liminfmes et limsupmes deux entiers, liminfmes $\leqslant$ limsupmes
la procédure renvoie une valeur de correct = 'n'
ainsi qu'un message d'erreur si coéff n'est pas compris
entre liminfmes et limsupmes.

- recopiedau : cette procédure recopie le contenu du fichier schéma2 dans le fichier (schéma). Cette copie est nécessaire car c'est toujours sur (schéma) que l'on travaille.

Spécification de recopiedau

Soit schéma, schéma2, fichiers séquentiels de flèches
la procédure recopie les articles de schéma2 dans schéma.

2) retflèche : est une procédure qui permet de retirer une flèche du schéma, c'est-à-dire de retirer un article du fichier.

Spécification de retflèche

Soit schéma un fichier séquentiel de flèches,
la procédure, après avoir acquis interactivement la flèche à retirer, renvoie un fichier séquentiel de flèches identique à celui entré mais dont 0 ou 1 article aura été retiré (selon que l'article à retirer est validé ou pas).
De plus, la procédure met à jour la liste des arcnuls.

Le principe est le suivant :

Soit à retirer l'arc d'origine A et d'extrémité B où A et B ont été validés comme étant des sommets existants dans le schéma.
On parcourt le fichier (schéma)

- si l'article $\neq$ (A,B) on le recopie dans le deuxième fichier
- si l'article = (A,B) on ne le recopie pas
- si on arrive en fin de fichier sans avoir rencontré d'article d'origine A et d'extrémité B, on en déduit que cet article n'existait pas, et on en avertit l'utilisateur.

Si tout se passe bien, on a donc dans (schéma2) la copie de (schéma) sans l'article à retirer. C'est bien ce que l'on voulait.
On recopie (schéma2) dans (schéma) puisque c'est toujours sur (schéma) que l'on travaille.

Les outils de niveau 1 utilisés ici sont :

- listarcnul : car si on retire un arc de délai nul, il faut également le retirer de la liste des arcs nuls; ceci se fait via la procédure de gestion de cette liste : listarcnul.

- validsommet : vérifie que A et B sont des sommets du schéma. Si ce n'était pas le cas, la procédure retourne un message d'erreur à l'utilisateur, et le procédure retflèche se termine ainsi.

- recopiedau : effectue la copie de (schéma2) dans (schéma).

3) retsom : est une procédure qui permet de retirer un sommet A, par exemple, du graphe et par là même, toutes les flèches dont l'origine ou dont l'extrémité est égale à A.

Spécification de retsom

Soit schéma un fichier de flèches et
Soit somval une liste de sommets
la procédure obtient interactivement le nom d'un sommet à retirer, et renvoie un message d'erreur si ce sommet n'appartient pas à la liste somval et modifie la liste somval en y retirant ce sommet s'il s'y trouve.
Le procédure retire également du fichier schéma les arcs dont l'origine ou l'extrémité égale ce sommet et met à jour, conséquemment, la liste des arcs nuls.

Le principe est le suivant :

Soit à retirer le sommet A, A ayant été validé comme étant un sommet du graphe.
On parcourt le fichier (schéma) et on recopie dans (schéma2) tout article dont l'origine et l'extrémité diffèrent de A. Les autres articles ne sont pas recopiés.
Le fichier (schéma2) est ensuite recopié dans (schéma).
Et on efface le sommet de la liste des sommets.

Les outils de niveau 1 utilisés ici sont donc :

- listsomval : procédure gérant la liste des sommets, dont on retire le sommet A.
- listarcnul : utilisée ici car tout arc de délai nul retiré du fichier schéma doit également l'être de la liste des arcs nuls
- validsommet : si le sommet à retirer n'existe pas, la procédure validation retourne un message d'erreur à l'utilisateur et la procédure retsom s'achève.

4) modflèche : est une procédure qui permet de modifier les valeurs associées à un arc.

Spécification de modflèche

Soit schéma un fichier séquentiel de flèches,
la procédure acquiert interactivement la flèche à retirer, et renvoie un message d'erreur si la flèche n'existe pas dans schéma, sinon renvoie le fichier schéma dans lequel 0 ou 1 article aura été modifié.
De plus, la procédure met à jour la liste des arcs nuls.

Le principe est le suivant :
Soit à modifier les valeurs attachées à l'arc d'origine A et d'extrémité B où A et B ont été validés comme étant des sommets du graphe.

On parcourt le fichier à la recherche de l'article représentant l'arc (A,B)

- tout autre article est recopié tel quel dans schéma2
- l'article représentant l'arc (A,B) est recopié avec les nouvelles valeurs du délai et du coéfficient qui auront été demandées et validées.
- Si on arrive en fin de fichier sans avoir trouvé d'article d'origine A et d'extrémité B, on en déduit qu'un tel arc n'existait pas dans le graphe. On en avertit l'utilisateur et la procédure se termine.
- Dans les autres cas, on recopie le fichier (schéma2) dans le fichier (schéma) sauf si les nouvelles valeurs introduites sont égales aux anciennes valeurs, car alors (schéma2) = (schéma) et il est inutile d'effectuer une copie.

Les outils de niveau 1 utilisés ici sont :

- listarcnul : - Si le nouveau délai est nul et que l'ancien délai était strictement positif, on utilise la procédure listarcnul pour rajouter ce nouvel arc à la liste.
  - Si l'ancien délai était nul et que le nouveau ne l'est pas, on retire l'arc de la liste par cette procédure.

- validsommet : si un des sommets A ou B n'existe pas, la procédure validsommet retourne un message d'erreur à l'utilisateur et le procédure modflèche se termine.

- validtemps et validmesure : puisque ni le délai, ni le coéfficient ne peuvent être quelconques.

- recopiedau : pour la copie de (schéma2) dans (schéma).

5) modsom : cette procédure permet de modifier la valeur attachée à un sommet.

Spécification de modsom

Soit somval une liste de sommets et de leurs valeurs respectives
la procédure acquiert interactivement le nom du sommet dont la valeur doit être modifiée, et renvoie un message d'erreur si ce sommet n'appartient pas à la liste, et sinon modifie la valeur de ce sommet dans la liste.

Le principe est le suivant :

Etant donné un sommet A validé comme étant un sommet de graphe,
dont on doit modifier la valeur,
on recherche ce sommet dans la liste,
on modifie sa valeur qui peut être quelconque.

Outils utilisés :

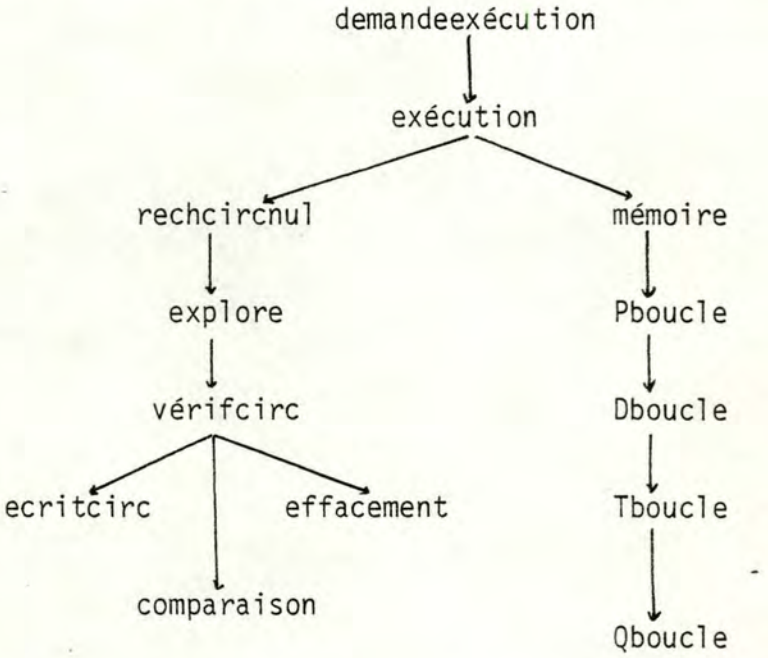
-listsomval : procédure de gestion de la liste du sommet

-validsommet: si le sommet A n'existe pas, l'utilisateur en est averti et la procédure modsom se termine.

D. LE MODULE D'EXECUTION DE L'ALGORITHME DE RECURRENCE.

Cette exécution s'effectue autant de fois que l'utilisateur le désire, c'est pourquoi la procédure exécution s'inscrit dans la procédure demandeexécution.

Le module présente la structure suivante :



La procédure exécution se charge de vérifier que les conditions d'éxécution de l'algorithme de récurrence sont réunies. C'est-à-dire, on vérifie que le schéma est non vide, et qu'il n'existe pas de circuit de délai total nul.

Spécification de la procédure exécution:
Soit schéma un fichier de flèches,
Soit nart un entier ≥ 0, la procédure renvoie les résultats 0,1 ou plusieurs simulations sur ce schéma et renvoie un message d'erreur si nart = 0 ou si le schéma comporte un circuit de délai total nul. Dans ce dernier cas, le message donnera les éléments constituant ce circuit.

1) La procédure rechcircnul : se charge de repérer les circuits de délai total nul du graphe, elle renvoie donc une valeur de indiccirc = 1 si un tel circuit existe et, de plus, elle signale à l'utilisateur l'existence d'un circuit de délai total nul en fournissant la liste des sommets qui forment ce circuit.

Pour cela, on dispose de quatre listes :

1) arcnul : la liste des arcs de délai nul qui aura été constituée lors de l'introduction des données et éventuellement modifiée au cours des diverses modifications effectuées sur le schéma.

2) circnul : est une liste des sommets, descendant du premier élément de cette liste (choisis selon un processus défini ultérieurement)par un chemin de délai total nul.
Chaque fois qu'on ajoute un sommet à cette liste, on le compare à tous ceux qui le précèdent sur cette même liste.
S'il y a égalité entre ce sommet et le ième élément, par exemple, on en déduit qu'il existe un circuit de délai total nul composé de tous les sommets de la liste compris entre le ième et le dernier élément y compris.

3) elmt : est une liste dans laquelle on range les sommets constituants les circuits trouvés, les uns à la suite des autres.

4) En correspondance avec la liste précédente, on a une liste marque qui contient les longueurs des circuits placés dans la liste elmt. Ainsi, le ième nombre de cette liste indique-t-il la longueur du ième circuit placé dans la liste elmt.

Le principe de recherche des circuits de délai total nul est alors le suivant :

On dispose d'une liste des arcs de délai nul et on prend successivement chacun des arcs de cette liste pour le placer dans la liste circuit.

Soit le ième arc de la liste arcnul, avec ori_i = origine de ce ième arc

et ext_i = extrémité de ce ième arc

On a donc la liste circnul

$$circnul = (ori_i , ext_i).$$

Pour poursuivre la construction de cette liste, on recherche dans arcnul
l'(es) arc(s) dont l'origine vaut ext_i (procédure explore)
Soit $ori_j = ext_i$
On place donc ext_j dans circnul

$$circnul = (ori_j, ext_i, ext_j)$$

On vérifie qu'aucun sommet existant déjà dans cette liste n'est égal à ext_j, auquel cas on aurait un circuit.

Si donc $ext_j \neq ori_i$ et $ext_j \neq ext_i$, on recommence à rechercher dans la liste des arcs nuls celui (ceux) dont l'origine est ext_j.

Soit $ori_k = ext_j$, on place alors ext_k dans circnul

$$circnul = (ori_i, ext_i, ext_j, ext_k)$$

On suppose maintenant que $ext_k = ext_i$, par exemple. On a donc repéré un circuit nul : $(ext_i, ext_j, ext_k = ext_i)$.

Qu'en fait-on ? On le sauve en le recopiant dans la liste elmt, qui contient déjà les circuits nuls différents qu'on a pu trouver. Et on stocke la longueur de ce nouveau circuit dans la liste marques.

La procédure comparaison intervient à ce niveau, elle compare le nouveau circuit à tous ceux de même longueur qui existent déjà dans elmt.
Ceci permet d'éviter d'avoir autant de fois le circuit qu'il n'y a d'éléments qui le composent. Par exemple, si on a déjà trouvé le circuit (A, B, A), on ne prendra plus le circuit (B, A, B).

Si la comparaison est positive, on efface alors le circuit de cette liste elmt, et on efface sa longueur de la liste marque (procédure effacement).

Si la comparaison est négative, on signale à l'utilisateur l'existence de ce circuit dont la procédure écriture affiche les éléments.

On ne retire pas ce circuit de la liste elmt, car il doit pouvoir servir de point de comparaison pour les prochains circuits nuls qu'on trouverait.

Tout ceci fait, on retire ext_k de la liste circnul, et on poursuit le travail, en cherchant un (des) autre(s) arc(s) nul(s) d'origine égale à ext_j.

S'il n'y en a plus, on retire ext_j de la liste et on poursuit par la recherche de(s) arc(s) de délai nul dont l'origine vaut ext_i,

et ainsi de suite.

Quand tous les chemins issus de l'arc (ori_i, ext_i) ont été exploités, on passe à l'exploitation de l'arc suivant dans la liste des arcs nuls, et on recommence donc le même procédé avec

$$circnul = (ori_{i+1}, ext_{i+1}).$$

Les outils utilisés par cette procédure rechcircnul sont :

listarcnul : car on va chercher dans la liste des arcs nuls, les arcs que l'on exploite à la recherche de circuits de délai total nul.

listcircnul : gère la liste circnul et les accès à celle-ci

gestelmt : " " elmt " "

gestmarq : " " marq " "

NB .: Il n'est malheureusement pas possible de construire facilement une unique procédure de gestion de toutes les listes car celles-ci sont constituées d'éléments de formats différents.

2) La procédure mémoire est donc celle qui, pour un cvd qu'elle demande et qu'elle valide comme étant un sommet de graphe, pour un temps imparti, qu'elle demande et qu'elle valide pour une impulsion donnée quelconque, exécute l'algorithme de récurrence présenté en 1ère partie.

La matrice D ne peut être implémentée en tant que matrice, puisque le PASCAL n'admet pas les tableaux à bornes variables.
Or la matrice D possède TI lignes, avec TI variant d'une exécution de la procédure mémoire à l'autre. La matrice possède également NC colonnes où NC est inconnu a priori. On pourrait approcher NC par NS, le nombre de sommets du graphe, mais ce ne serait pas suffisant puisqu'un sommet peut se présenter plusieurs fois sur une même ligne.
Dès lors, trois solutions se présentent :

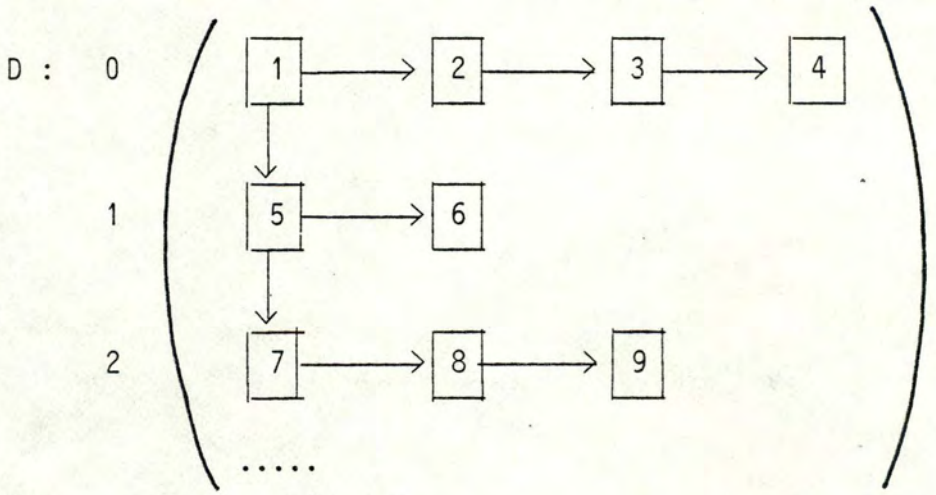
a.- mettre au point une procédure d'exception lorsque C(I) dépasse la valeur NS.
Ceci risque d'être complexe et peu pratique.

b.- prévoir un maximum absolu du nombre de colonnes que D peut avoir, c'est-à-dire du nombre d'éléments qu'il peut y avoir sur une ligne de D.
Dans ce cas, on risque de créer une matrice énorme et quasi vide.

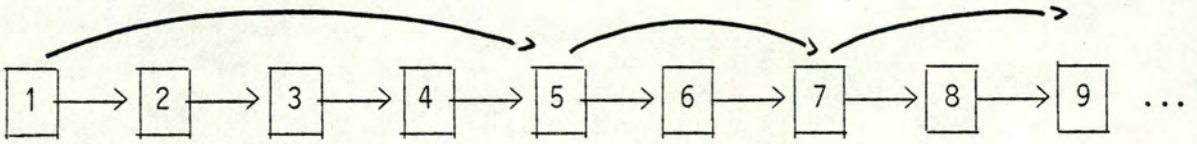
c.- implémenter la matrice par pointeurs, comme pour les listes, et établir une procédure de traduction de l'écriture D(i,j) vers l'élément adéquat dans la liste qui représente la matrice.

C'est cette 3ème solution que l'on adopte ici. On constitue une seule liste, avec deux types de pointeurs différents : un pointeur vers l'élément suivant sur la ligne, et un pointeur vers l'élément suivant sur la colonne, ceci uniquement pour la première colonne puisque la matrice ne se parcourt jamais colonne par colonne, mais ligne par ligne.
Le pointeur colonne sert uniquement à passer d'une ligne à l'autre.

Ainsi D peut-elle être représentée de la façon suivante :



Ce qui peut également se représenter de cette façon :



La procédure gestmat gère cette liste et les accès à celle-ci.

La procédure positionner (i,j) est également un outil de niveau 1 qui effectue la traduction de l'écriture D(i,j) vers l'élément adéquat de la liste.

La procédure mémoire comprend les 4 boucles décrites en 1ère partie.
Il reste à prouver que la 4ème boucle est correcte.

Reprenons sa spécification :

Soit X' un sommet du graphe,

Soit I' un entier tel que $0 \leqslant I' \leqslant TI$

et i' un entier quelconque

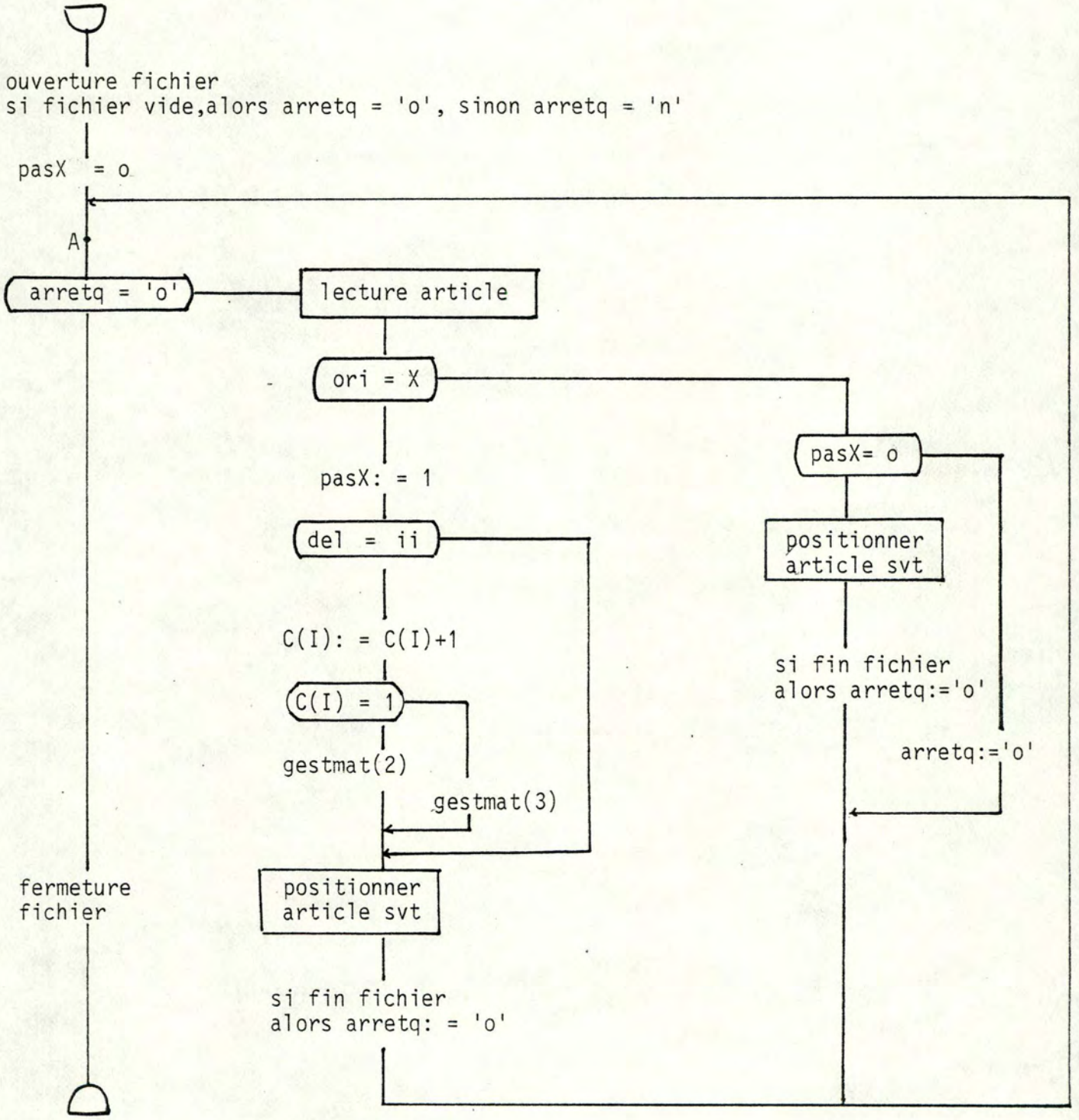
Soit C' nombre entier $\geqslant 0$

nb' le nombre d'arcs de délai i' issus de X'

L'exécution de la boucle avec X = X', i = i', I = I' et C' = C(I')

- établit C(I'): = C' + nb'
- place sur D(I', C'+1) ... D(I', C'+nb') les extrémités des arcs de délai i' issus de X'
- ne modifie pas les valeurs de i et I, ni les valeurs des autres éléments de C et de D.

4ème boucle.


ouverture fichier
si fichier vide,alors arretq = 'o', sinon arretq = 'n'
pasX = o
A
arretq = 'o'
lecture article
ori = X
pasX: = 1
del = ii
C(I): = C(I)+1
C(I) = 1
gestmat(2)
gestmat(3)
positionner article svt
si fin fichier alors arretq: = 'o'
fermeture fichier
pasX= o
positionner article svt
si fin fichier alors arretq:='o'
arretq:='o'

1ère étape de la méthode de l'invariant.

A chaque entrée dans la boucle, les variables vérifient I_A

où I_A :

1) Arretq = '0' $\Longrightarrow$ tous les articles dont l'origine est X ont été traités.

2) Pasx = 0 $\Longleftrightarrow$ aucun article d'origine X n'a déjà été traité.

3) $C' \leqslant C(I') \leqslant C' + nb'$

4) D(I', C'+1) ... D(I', C(I')) représentent les extrémités des articles de délai i' d'origine X', qui ont déjà été traités.

- lors du premier passage:

1) Si le fichier est vide $\Rightarrow$ arretq = '0' et tous les articles dont l'origine est X ont été traités
car le fichier est vide, donc a fortiori, il n'existe pas d'articles d'origine X.

2) Evident.

3) au départ, la boucle s'exécute avec C' = C(I')
et puisque nb' $\geqslant$ 0, on a bien $C' \leqslant C(I') \leqslant C' + nb'$

4) D(I', C'+1) ..... D(I', C') = $\emptyset$, comme aucun article n'a été traité, cela représente bien les extrémités des articles de délai i' et d'origine X' qui ont déjà été traités.

- lors du nème passage :

I_A est supposé vérifié, et donc

1) Arretq = '0' $\Rightarrow$ tous les articles d'origine X ont été traités
mais arretq < > '0' , sinon il ne pourrait y avoir de passage suivant.

2) pas X = 0 $\Leftrightarrow$ aucun article d'origine X n'a déjà été traité.

3) $C' \leqslant C_n(I') \leqslant C'+nb'$.

4) $D(I', C'+1)$ ..... $D(I', C_n(I')$ représentent les extrémités des articles de délai i', d'origine X', qui ont déjà été traités.

- La boucle s'exécute dans ces conditions, il faut voir que I_A est toujours vérifié lors du passage suivant

1')si arretq = '0', cela signifie
soit qu'on est arrivé en fin de fichier
et donc, forcément, tous les articles d'origine X auront été traités

soit qu'ori $\neq$ X et que pas X $\neq$ 0

or si pas X $\neq$ 0 $\Leftrightarrow$ au moins un article d'origine X a été traité par 2)

or les articles de même origine sont groupés, aussi dès qu'on lit un article d'origine $\neq$ X avec pasX $\neq$ 0, on sait qu'on a traité tous les articles d'origine X'.

2')pas X = 0 $\Leftrightarrow$ aucun article d'origine X n'a déjà été traité
par 2) on sait que c'était vrai avant l'exécution de la boucle ,
si après la boucle on a toujours pasX = 0, il faut prouver qu'aucun article d'origine X n'a été traité
or pasX ne varie que si ori = X, et donc que si l'on traite un article d'origine X
pour la même raison, si après la boucle pasX = 1, on aura bien qu'au moins un article d'origine X n'aura été traité.

3') $C' \leqslant C_{n+1}(I') \leqslant C' + nb'$

car $C_{n+1}(I') = C_n(I')$ si ori $\neq X$

ou si ori = X et del $\neq$ i

dans ces cas là on a évidemment 3') puisqu'on avait 3)

mais $C_{n+1}(I') = C_n(I')+1$ si ori = X et del = i

$C' \leqslant C_n(I') \leqslant C_n(I')+1 = C_{n+1}(I')$

d'autre part C'+nb' = C' + nombre d'arcs de délai i' issus de X'

$\geqslant$ C' + nombre d'arcs de délai i' issus de X'

qu'on a déjà trouvés

$\geqslant C_{n+1}(I')$

4') par 4) on sait que

$D(I', C'+1)$ ..... $D(I', C_n(I'))$ = extrémités des articles de délai i' issus de X', qu'on a déjà traités.

Si $C_{n+1}(I') = C_n(I')$ dans 4' est directement prouvé

Si $C_{n+1}(I') = C_n(I') + 1 \iff$ ori = X, del = i', on a donc trouvé un article d'origine X' et de délai i'

gestmat place l'extrémité de cet article en $C(I) = C_n(I)+1=C_{n+1}(I)$

dès lors, on a bien $D(I', C'+1)$ ... $D(I', C_{n+1}(I'))$ = extrémités des articles de délai i', d'origine X' qu'on a déjà traités.

2ème étape de la méthode de l'invariant.

Si la boucle se termine $\Longleftrightarrow$ arretq = '0'

$\overset{(1)}{\Longleftrightarrow}$ tous les articles d'origine X' ont été traités

$\overset{(2)}{\Longrightarrow}$ D(I', C'+1) ... D(I', C(I')) représentent les extrémités des arcs de délai i' issus de X'.

$\Longrightarrow$ C(I') = C' + nb *

$\Longrightarrow$ les spécifications sont vérifiées.

3ème étape de la méthode de l'invariant.

La boucle se termine après un nombre fini de tours car

arretq = '0', soit un tour après avoir parcouru tous les articles d'origine X' qui sont en nombre fini

soit en fin de fichier qui comprend un nombre fini d'articles.

Les outils utilisés sont :

- gestmat pour la gestion de la matrice et des accès à celle-ci
- positionner (I,J) permet de se positionner sur l'élément (I,J) de la matrice
- validsommet: pour valider le cvd comme existant
- validtemps : pour valider le TI.
- resultaffich : procédure d'affichage des résultats détaillés de la simulation
- recopaffich : procédure d'affichage des résultats de la simulation en récapitulant ceux-ci par délai et par sommet
- printdemande est un outil qui demande à l'utilisateur s'il veut ou pas garder les résultats de sa simulation dans un fichier résultat, ce qui lui permettra, par après, d'imprimer ces résultats.

Chaque impression de résultat est constituée de 4 parties :

1) affichage du schéma tel qu'il était au moment de la simulation
2) affichage des conditions de simulation
   c'est-à-dire valeur du cvd, de TI et de l'impulsion de départ
3) tableau récapitulatif des résultats
4) le détail des opérations qui ont permis d'établir le tableau précédent.

Les outils utilisés sont :

- gestmat pour la gestion de la matrice et des accès à celle-ci
- positionner (I,J) permet de se positionner sur l'élément (I,J) de la matrice
- validsommet: pour valider le cvd comme existant
- validtemps : pour valider le TI.
- resultaffich : procédure d'affichage des résultats détaillés de la simulation
- recopaffich : procédure d'affichage des résultats de la simulation en récapitulant ceux-ci par délai et par sommet
- printdemande est un outil qui demande à l'utilisateur s'il veut ou pas garder les résultats de sa simulation dans un fichier résultat, ce qui lui permettra, par après, d'imprimer ces résultats.

Chaque impression de résultat est constituée de 4 parties :

1) affichage du schéma tel qu'il était au moment de la simulation
2) affichage des conditions de simulation
   c'est-à-dire valeur du cvd, de TI et de l'impulsion de départ
3) tableau récapitulatif des résultats
4) le détail des opérations qui ont permis d'établir le tableau précédent.

E. MODULE DE PREPARATION ET TERMINAISON DES FICHIERS.

En fait, il existe deux façons de rentrer dans le programme :

- Soit on n'a pas encore constitué le fichier correspondant au schéma sur lequel on veut effectuer une simulation; dans ce cas, (schéma) et (fsommet) sont vides en entrée.
  On utilise alors la procédure introduction pour constituer le fichier (schéma).

- Si, lors d'une exécution précédente, on avait déjà constitué le fichier, on peut alors entrer à nouveau dans le programme avec (schéma) et (fsommet) non vides.

  Dans ce cas, on commencera par reconstituer la liste somval à partir du fichier (fsommet) par la procédure réhabilitation.

  Ensuite, on reconstitue la liste des arcs nuls, à partir du fichier (schéma) au moyen de la procédure reformation.

  Si un sommet, origine ou extrémité, rencontré dans le fichier (schéma) ne se retrouve pas dans la liste des sommets (fsommet), on demande leur valeur et on les y ajoute - (permet de créer des jeux de valeur différents).

  En fin de programme, la procédure sauvetage sauve dans le fichier (fsommet) la liste des sommets du graphe, ainsi que leurs valeurs respectives.

F. CARACTERISTIQUES DU LOGICIEL

On remarque que, grâce à sa construction hiérarchisée et modulaire, le programme présente des facilités de maintenabilité, réutilisabilité, portabilité, convivialité et fiabilité.
Elles sont décrites ci-dessous.

1) Maintenabilité.

. modifiabilité : celle-ci est rendue aisée du fait que toute modification est localisée.

Par exemple, une modification des messages d'erreur ne touche que le module de validation. Il ne faut donc pas modifier à chaque endroit du programme où une validation est exigée, mais uniquement dans le corps des procédures de validation.

Un autre exemple : une modification de présentation des arcs d'un schéma sera localisée à la procédure d'affichage de ceux-ci, plutôt que partout où un affichage est demandé.

. extensibilité : si d'autres opérations de traitement des données s'avéraient nécessaires, il suffirait de construire un nouveau module, puisque les données et leur gestion sont localisées.

2) Ré-utilisabilité.

Si on conçoit des applications similaires, il est possible de réutiliser les composants du programme dans un contexte différent. Ainsi, un programme de recherche des villes accessibles en un temps donné via un réseau ferroviaire, par exemple, peut-il être construit à partir des mêmes composants que ceux du programme.

On aurait un fichier schéma dont les articles représenteraient les trajets de trains.

l'origine de l'arc devient la ville de départ

l'extrémité de l'arc devient la ville de destination

le délai de l'arc représente le temps nécessaire pour atteindre la ville de destination à partir de celle de départ

la mesure de l'arc pourrait être, par exemple, le prix de ce trajet.

Le nom des villes pouvant comporter plus de lettres que ceux des sommets des schémas économiques, il faudra modifier les constantes nblettres et blanc. De même, la limite inférieure de la mesure serait 0, puisqu'un prix ne peut être négatif, on modifie les constantes liminfmes et limsupmes.

L'algorithme de récurrence permettrait de donner la liste des villes qu'on pourrait atteindre à partir d'une ville de départ (cvd) dans un délai inférieur ou égal à un temps imparti donné, ainsi que les prix des trajets relevés, si toutefois ceux-ci se multiplient. Comme ils s'additionnent, il faudra modifier gestmat(2) et (3) qui s'occupe d'établir une mesure à chaque nouvel élément.
Le module introduction permettrait de construire le réseau ferroviaire étudié.
Le module de modification permettrait de modifier ce réseau.
Par exemple, grâce à la procédure modflèche, on pourrait modifier le temps nécessaire pour aller d'une ville à l'autre si un nouveau train, plus rapide était mis en fonctionnement sur cette voie.
On pourrait modifier le prix du trajet, en conséquence, par exemple.

Si aucune valeur n'est attachée aux sommets dans ce contexte, il suffirait de modifier la procédure de gestion de la liste somval ou, mieux encore, on pourrait supposer que toutes les valeurs existent déjà; il suffirait donc, dans le corps du programme, de mettre la variable valeurexiste à 1 partout.

3) Portabilité.

Grâce à la structure modulaire, on a localisé dans des composantes spécifiques tout ce qui dépend d'une configuration particulière.

Ainsi, la gestion des listes rendue nécessaire par le fait que le PASCAL sur le DEC n'accepte pas de tableaux à bornes variables est-elle localisée.
Si, pour une autre configuration, le PASCAL admettait de tels tableaux, il suffirait de modifier les procédures de gestion des listes pour en faire des procédures de gestion de matrices ou de vecteurs.

En bref, cette structure permet de localiser, autant que possible, tout ce qui peut changer dans le temps, et de localiser pour un mieux les décisions de conception.

Il reste que, dans ce programme, la décision de conception qu'était le choix d'une structure séquentielle, justifiée précédemment, ne soit pas tellement localisée.
Puisqu'elle intervient dans tous les modules principaux du programme, c'est-à-dire les quatre modules du niveau 2.
Un changement d'orientation dans ce domaine provoquerait des modifications importantes dans le programme. Cela ne veut pas dire que ces modifications seraient en elles-mêmes difficiles, mais elles ne seraient pas localisées dans le programme. Pour pallier cela il aurait fallu former un module de gestion du fichier et des accès à celui-ci, comme on l'a fait pour les listes.

4) Convivialité.

Il existe un module d'assistance à l'utilisateur : help, isolé dans une composante bien spécifique plutôt qu'éparpillée partout où une aide peut s'avérer utile.

5) Fiabilité.

. validité : le programme apparaît conforme aux spécifications du problème (cf. application dans la 3e partie).

. robustesse : par des procédures de validation, et d'autres moyens, on a tâché de construire un programme tolérant aux fautes qui lui sont indépendantes. On a prévu des mécanismes de récupération d'erreur.

Ainsi, si un utilisateur fournit un schéma contenant un circuit de délai total nul, le programme lui proposera de modifier son schéma. Ou,si un utilisateur, par inadvertence, veut entrer une boucle, le programme lui envoie un programme d'erreur et lui propose de fournir une nouvelle extrémité à cet arc.

. testabilité : la structure modulaire et hiérarchisée permet de tester les modules et outils indépendamment du contexte d'application. Principalement, si la relation d'hiérarchisation est du type "utilise".

G. MODE D'EMPLOI DU PROGRAMME.

Le programme commence par demander les noms des trois fichiers :

1) Schéma : il faut alors donner

- soit le nom du fichier, vide mais existant, qui contiendra le schéma que l'utilisateur entrera.
  Cela signifie qu'il faudra éventuellement se créer un fichier vide avant de pouvoir exécuter le programme.

- soit le nom du fichier qui contient le schéma, déjà construit, sur lequel on effectue la(es) simulation(s).

Le fichier devra être séquentiel.

2) fsommet : l'utilisateur donnera le nom du fichier qui contient les sommets de son schéma et les valeurs de ces sommets.

L'utilisateur peut éventuellement entrer un fichier vide, auquel cas il construira un nouveau jeu de valeur pour les sommets de son schéma.

3) Résultat : est un fichier texte qui permet de garder les résultats des diverses simulations. L'utilisateur entrera donc le nom du fichier dans lequel il veut voir sauver les résultats de ses simulations.

Exemple : exec
link loading

schéma : revenu . seq

fsommet : somrev . seq

résultat : result . seq ou result . txt.

Lors de l'exécution du programme, celui-ci dirige l'utilisateur dans ses simulations, dans ses choix et dans ses modifications.
Chaque fois qu'une réponse alphabétique à une question posée par le programme est requise, l'utilisateur peut taper help.
Un message indicateur se présentera alors à l'écran.

TROISIEME PARTIE

APPLICATION DU PROGRAMME
AU PROBLEME
DE LA REDUCTION SALARIALE

==========

Pour appliquer le programme, on utilise le modèle économique représentant les effets sur l'économie belge d'une réduction salariale. Ce modèle a été construit à partir du travail de C.Debatty et C.Ledouble dans le cadre du séminaire d'économie publique, année académique 82-83.

A. EXPLICATION DU PROBLEME.

Le gouvernement se propose de limiter la croissance des salaires nominaux (salaires en francs courants) en 1982 et 1983.
Il s'agit de suspendre momentanément le mécanisme d'indexation automatique des salaires.
Conformément à la loi du 1er mars 1977, chaque fois que l'indice-pivot varie de 2 %, les rémunérations sont recalculées à la hausse.
Cette adaptation est appliquée à partir du deuxième mois qui suit le mois durant lequel l'indice a atteint le chiffre justifiant une modification.
L'Arrêté Royal des pouvoirs spéciaux du 26 février 1982 a suspendu ce mécanisme, du moins pour les salaires dépassant le niveau minimum garanti. Il a prévu d'attribuer aux salariés une augmentation forfaitaire de leurs rémunérations égale à 2 % du salaire minimum, chaque fois que l'indice-pivot varie de 2 %. Cela ne modifiait donc rien pour les détenteurs de bas revenus mais cela freinera l'indexation des revenus plus élevés.

Ce mécanisme d'indexation forfaitaire ne démarra qu'au mois de mai pour ceux dont le salaire était supérieur au salaire minimum garanti.

Le principe de la modération salariale à appliquer en 1983 fut quelque peu différent. Il a prévu d'octroyer encore deux augmentations forfaitaires et de revenir ensuite au système traditionnel d'indexation des salaires, mais avec deux modifications :

1°) Les salaires n'ont été effectivement indexés que quatre mois après que l'indice-pivot ait été dépassé. Ceci ralentit le rythme des indexations.

2°) Une révision de la composition de l'indice des prix à la consommation eut lieu à partir du 1er juillet 1983.

On obtient (cf. travail Debatty-Ledouble) les pourcentages de réduction des salaires réels suivants :

1982 : 1,89 %

1983 : 5,509 %

On avait un revenu salarial annuel en 1981 = 495.372 FB

et donc un revenu salarial total = revenu salarial annuel x nombre de salariés

= 495.372 x 3.687.436

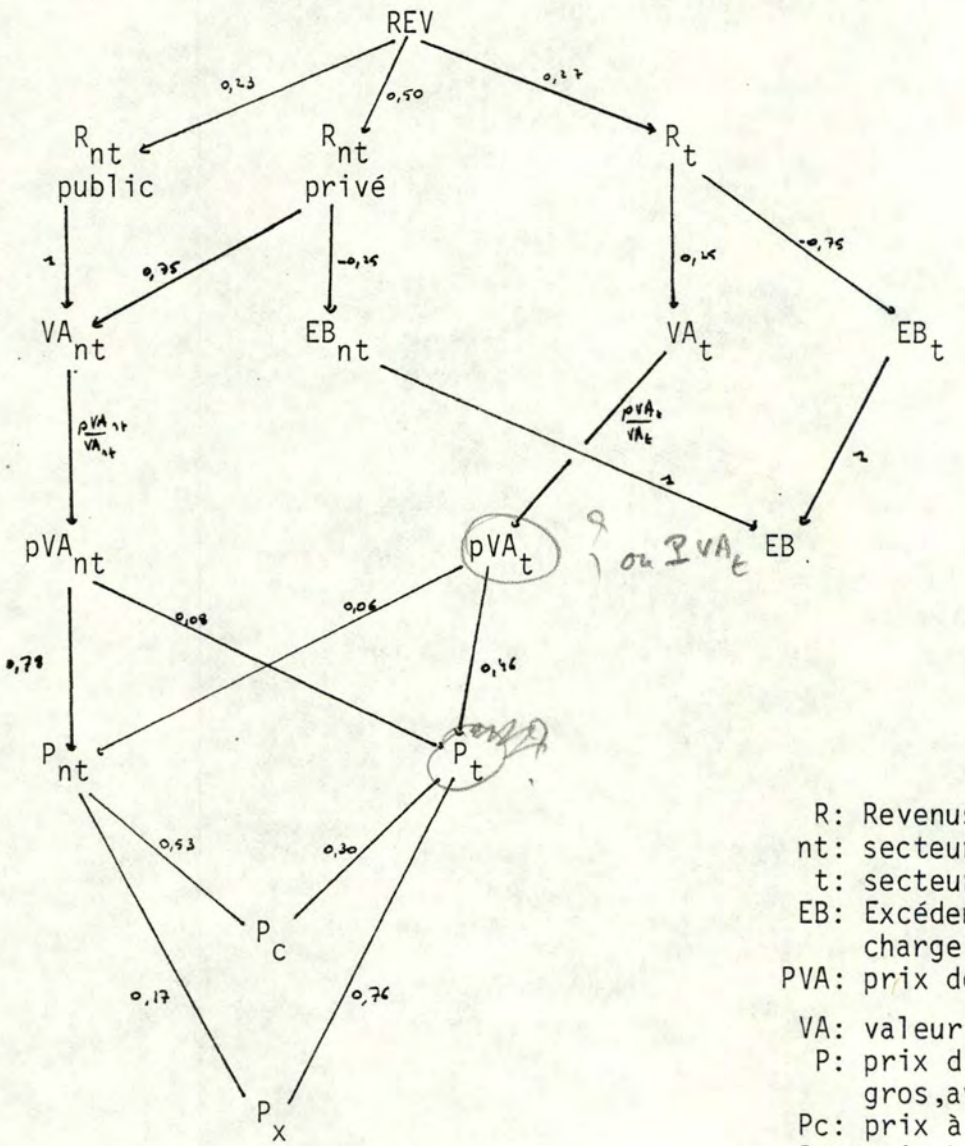
= 1.826,652 Mds

La variation de revenu provoquée par cette nouvelle réforme sera donc égale à (-5,5 %) (1.826,652) Mds, soit -100,630 Mds.
Ceci représente donc l'impulsion de départ fournie sur le centre de variation de départ, à savoir REV, les revenus salariaux.

On représente les différents effets de cette réduction salariale à l'aide des schémas économiques suivants. Ces schémas seront ensuite regroupés en un unique schéma qui servira de modèle à la simulation.

Remarque : On suppose que toutes les influences sont subies après un délai nul.

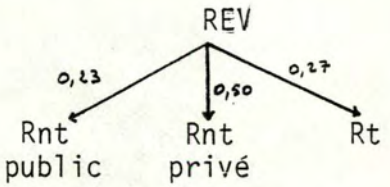
Schéma 1 :



R: Revenus
nt: secteur abrité
t: secteur exposé
EB: Excédent brut (cad pdt-charge, avant impôt)
PVA: prix de la valeur ajoutée
VA: valeur ajoutée
P: prix d'output (cad prix de gros, avant distrib.)
Pc: prix à la consommation
Px: prix à l'exportation

La réduction du salaire moyen en termes réels entraîne une réduction des rémunérations totales dans les secteurs abrités et exposés.
De plus, étant donné qu'on ne peut appliquer aux pouvoirs publics un raisonnement en termes de surplus brut d'exploitation, on isole au sein du secteur abrité ce qui est secteur privé de ce qui est secteur public.

D'où le schéma

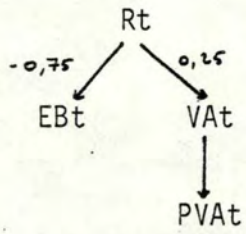


Les coéfficients des arcs du schéma proviennent du tableau suivant :

branche d'activité	nombre de personnes occupées	revenu annuel moyen	rémunération totale
secteur exposé	988.448	495.372	489,649 Mds soit 26,8% du total
secteur abrité			
- public	864.675	495.372	428,335 Mds soit 23,45%
- privé	1.834.313	495.372	908,667 Mds soit 49,74%
TOTAL	3.687.436	495.372	1.826,651 Mds

On suppose que la réduction des rémunérations est répartie de la façon suivante :

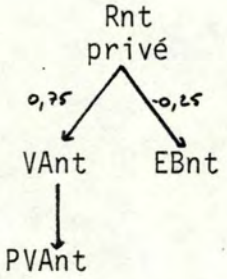
- pour le secteur exposé



3/4 de la réduction sont affectés à l'accroissement de l'excédent brut

et 1/4 de la réduction est affecté à la baisse du prix de la V.A.

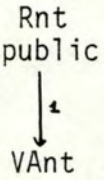
- pour le secteur abrité, privé



1/4 de la réduction est affectée à l'accroissement de E.B.

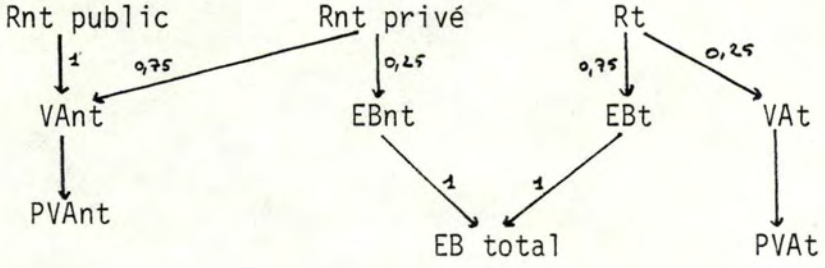
et 3/4 de la réduction sont affectés à la baisse du prix de la VA

- pour le secteur abrité, public



la totalité de la réduction est affectée à la baisse de la VA

Tout ceci se regroupe dans le schéma suivant :



On remarque que les flèches allant de EB_{nt}, EB_t vers EB_{total} ne sont pas vraiment identiques aux autres flèches car elles ne représentent pas une influence à proprement parler, mais une sommation.

$$EB_{total} = EB_{nt} + EB_t.$$

Il est cependant évident que si EB_t varie d'une quantité ΔEB_t, EB_{total} variera de la même quantité, c'est pourquoi on pourra également voir ces flèches comme la représentation d'une influence.

La variation globale des prix de l'output peut être décomposée comme suit :

- pour les biens du secteur ouvert :

$$\overset{\circ}{P}_t = 0{,}465\ \overset{\circ}{pVA}_t + 0{,}0845\ \overset{\circ}{pVA}_{nt} + 0{,}4505\ \overset{\circ}{P}_m$$

où le symbole $^\circ$ indique qu'on travaille en accroissements

où P_m est une variable exogène qui n'entre pas dans la composition du schéma représentant les prix à l'importation.

- pour les biens du secteur abrité :

$$\overset{\circ}{P}_{nt} = 0{,}7825\ \overset{\circ}{pVA}_{nt} + 0{,}0658\ \overset{\circ}{pVA}_t + 0{,}517\ \overset{\circ}{P}_m$$

- pour les biens exportés et consommés :

$$\overset{\circ}{P}_x = 0{,}7609\ \overset{\circ}{P}_t + 0{,}1702\ \overset{\circ}{P}_{nt} + 0{,}0689\ \overset{\circ}{P}_m$$

$$\overset{\circ}{P}_c = 0{,}3042\ \overset{\circ}{P}_t + 0{,}5281\ \overset{\circ}{P}_{nt} + 0{,}1677\ \overset{\circ}{P}_m$$

On obtient ainsi la fin du schéma 1 :

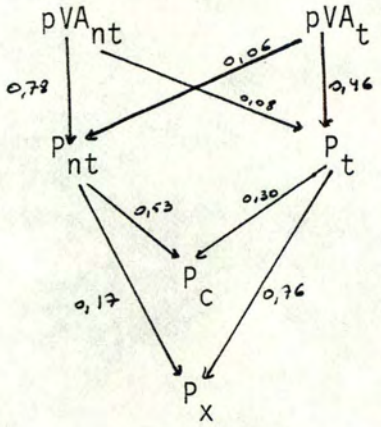
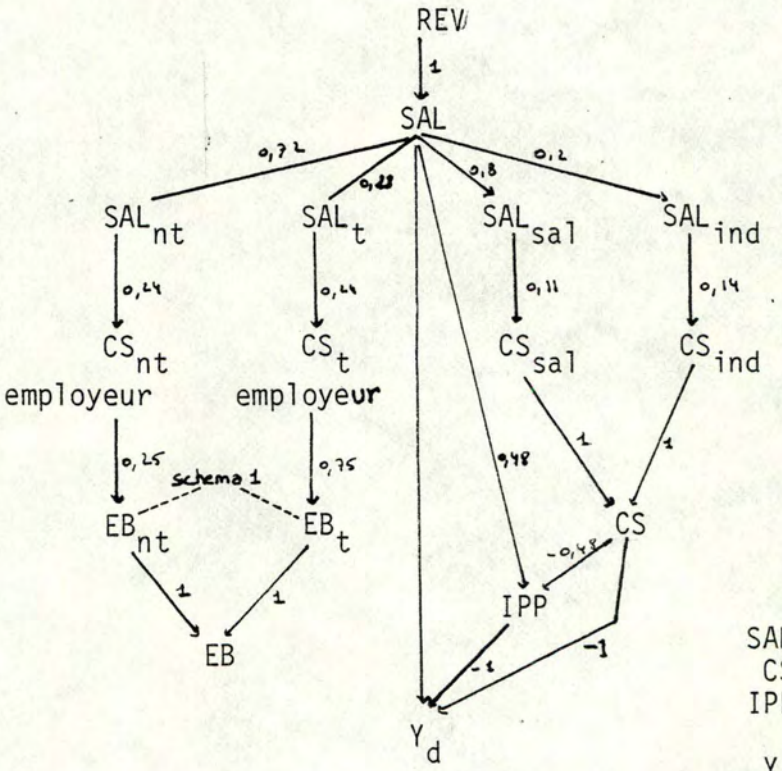


Schéma 2.



SAL: salaires
CS: cotisation salariale
IPP: impôts s/personnes physiques
Y_d: revenu disponible
sal: salariés
ind: indépendants

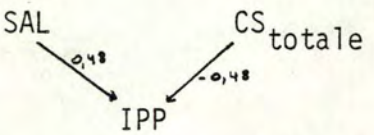
La réduction des rémunérations totales provoquée par la diminution des salaires réels réduit la masse des revenus salariaux bruts du même montant.

$$R \xrightarrow{1} SAL$$

Si la masse des rémunérations diminue, la masse des impôts directs sur le revenu et les cotisations sociales à charge des travailleurs diminuent également.
Le taux marginal des revenus moyens est de 45,8 %, et si on y ajoute les additionnels communaux (pourcentage des impôts sur le revenu perçu pour le compte des communes), il est de 45,8 % x 1,06 % = 48,548 %.

Ainsi $\Delta IPP = 0{,}485\ (\Delta SAL - \Delta CS_{totale})$.

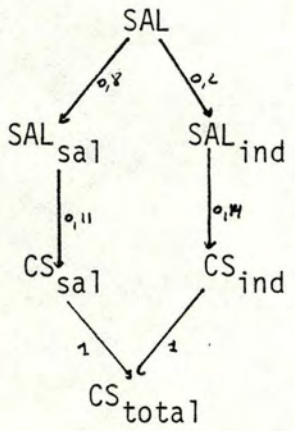
Ce qui se visualise de la façon suivante :



De plus, on tient compte du fait que le gouvernement désire également soumettre les indépendants à la modération des revenus.
On suppose alors que les indépendants et que les titulaires de professions libérales perçoivent un revenu mensuel moyen égal à celui des salariés. Cette hypothèse sous-estime bien entendu les revenus des indépendants.
La répartition de la population peut se faire ainsi : 80 % de salariés et 20 % d'indépendants et professions libérales.

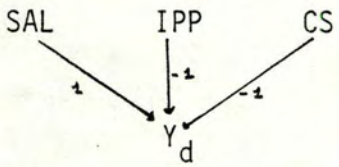
On regarde maintenant comment CS_{totale} a été obtenu.

Le taux de cotisation salariale est de 10,82 % pour les salariés et de 14,55 % pour les indépendants. C'est pourquoi on sépare l'influence SAL ——> CS en deux influences distinctes.



Le revenu disponible, quant à lui, Y_d = SAL - IPP - CS

et donc



D'autre part, pour les secteurs abrités et exposés, on peut considérer que la réduction des salaires a deux impacts.
Le premier est l'impact sur P_c et P_x décrit dans le schéma 1.
Le second provient de la diminution des cotisations sociales.
Selon la clé de répartition, le secteur exposé consacrera 3/4 de cette réduction à l'excédent brut d'exploitation, et le secteur abrité n'en consacre qu'un quart.
Que fait-on du 1/4 restant dans le secteur exposé et des 3/4 restants du secteur abrité ?
Le travail de Debatty-Ledouble n'en parle pas, aussi, pour rester cohérent avec leur modèle, on ne s'en préoccupera pas dans le cadre de la simulation.

On obtient le sous-schéma suivant qui représente ces faits :

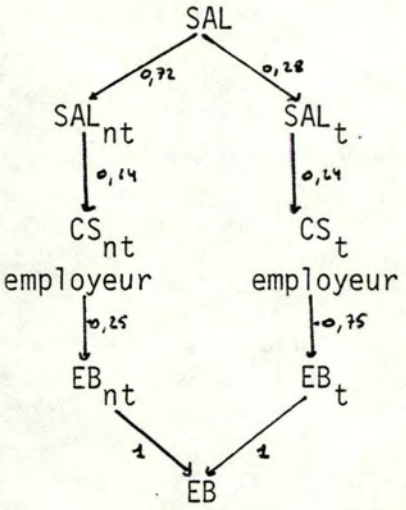
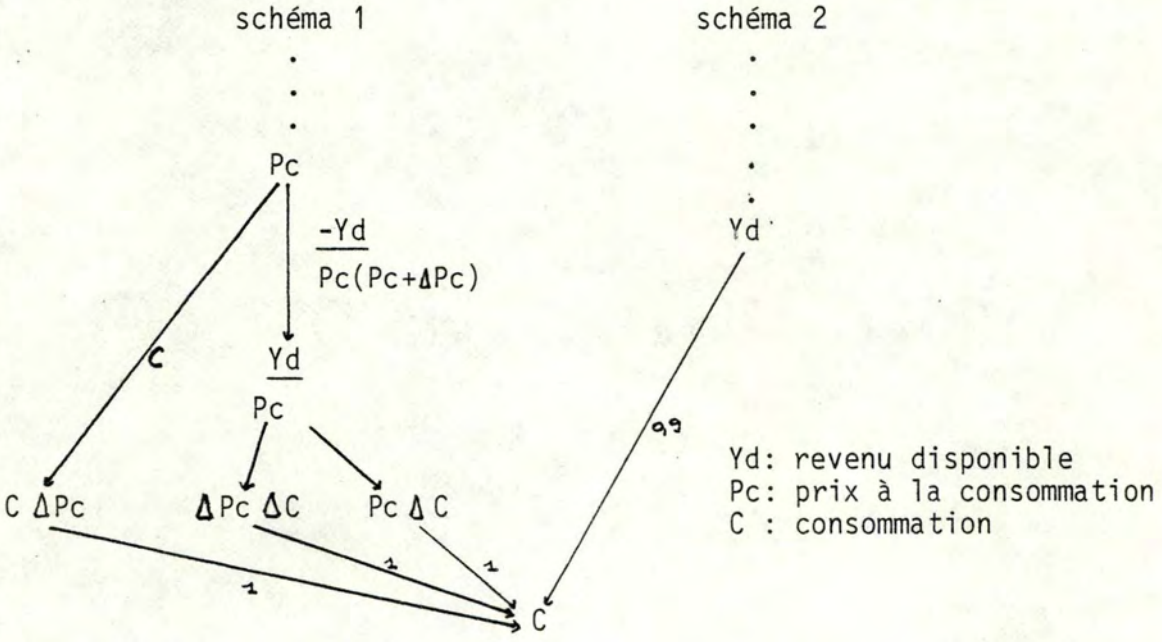


Schéma 3.



La diminution du niveau général des prix entraîne une augmentation du revenu disponible réel des ménages : $Pc \longrightarrow \frac{Yd}{Pc}$

La modification du revenu disponible réel se traduit par une variation de la consommation privée des ménages : $Yd \xrightarrow{0,9} C$.

On prend une élasticité à long terme de la consommation au revenu réel de 0,9. On utilise cette élasticité à long terme, malgré que tous les délais soient supposés nuls, car on ne dispose pas de l'élasticité à court terme.

La variation de la consommation induite par l'évolution du niveau général des prix se calcule de la façon suivante :

$$C\Delta Pc + Pc\Delta C + \Delta Pc\Delta C$$

où $C\Delta Pc$ est l'effet-prix à volume inchangé

$Pc\Delta C$ est l'effet-quantité à prix inchangé

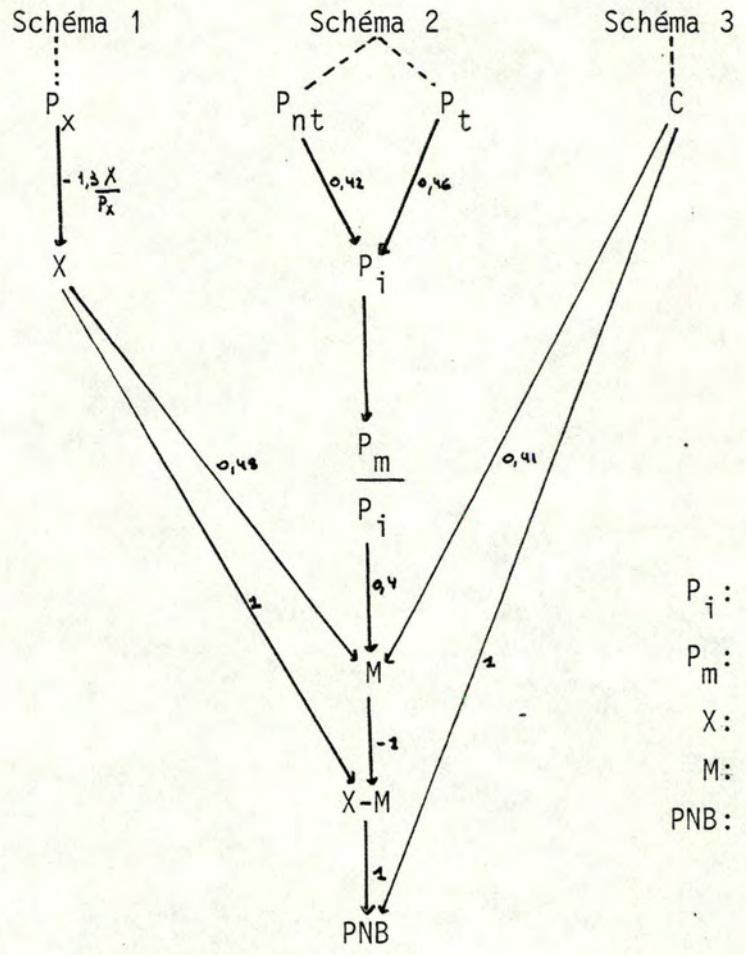
$Pc\Delta C + \Delta C\Delta Pc$ est l'effet-quantité au nouveau prix.

Un nouveau problème se pose ici, car on ne peut pas déterminer a priori les coéfficients à placer sur certains arcs.

Ainsi le coéfficient de l'arc $Pc \longrightarrow Yd/_{Pc}$ est-il égal à $- Yd / PC(Pc + \Delta Pc)$ qu'on ne peut déterminer a priori puisqu'on ne connaît pas PC. Une solution possible est d'établir une nouvelle flèche $Pc \xrightarrow{1} \Delta Pc$, ainsi, lors de l'exécution du programme, on trouverait au sommet ΔPc, de valeur initiale nulle, la valeur ΔPc qu'on pourra alors utiliser dans unedeuxième exécution pour établir le(s) coéfficient(s) manquant(s).

On agit de même pour connaître ΔC et établir les coéfficients des arcs $Yd/Pc \longrightarrow \Delta Pc\Delta C$, et $Yd/Pc \longrightarrow Pc\Delta C$ qui sont donnés sur le schéma général page 103.

Schéma 4.



P_i: prix intérieur

P_m: prix à l'importation

X: exportations

M: importations

PNB: produit normal brut

La diminution des prix à l'exportation entraîne, par hypothèse, une augmentation plus que proportionnelle des quantités exportées et, dès lors, provoque une augmentation de la valeur totale des exportations. Cette hypothèse est basée sur une élasticité de la demande d'exportation au prix des exportations égale à -1,3.

Ceci se représente par l'arc suivant :

$$P_X \xrightarrow[-1,3\,\frac{X}{P_X}]{} X$$

Etant donné que le volume des exportations augmente, la valeur des importations augmente également. On considère que le contenu total en importation des exportations est de 47,77 %.

D'où : $X \xrightarrow{0,48} M.$

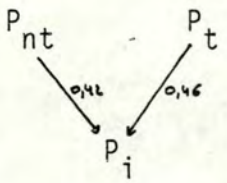
D'autre part, la modification des prix d'output dans les secteurs abrités et exposés de l'économie provoque une modification du niveau des prix intérieurs.

On utilise l'équation suivante :

$$\overset{\circ}{P}_i = 0,4584 \ \overset{\circ}{P}_t + 0,4178 \ \overset{\circ}{P}_{nt} + 0,1237 \ \overset{\circ}{P}_G$$

où P_G : prix des dépenses publiques, variable exogène, n'entre pas dans la composition du schéma.

Aussi visualise-t-on ces influences de la façon suivante :



et $P_i \longrightarrow P_{m/P_i}$ car sur base d'un prix à l'importation inchangé ($\overset{\circ}{P}_m = 0$), suite à la diminution des prix intérieurs, le rapport P_m/P_i augmente, les importations deviennent donc relativement plus chères.

A nouveau, on remarque qu'on ne peut donner a priori un coéfficient à cet arc. Il faut donc procéder de la façon présentée précédemment lors de la rencontre du même problème.

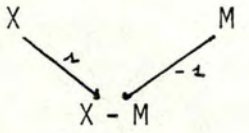
De plus, on prend une élasticité de la demande d'importation au prix relatif P_{m/P_i} égale à -0,4 :

$$P_{m/P_i} \xrightarrow[-0,4 \ \frac{MP_i}{Pm}]{} M$$

La variation de la consommation privée des ménages entraîne également une variation des importations. Prenons comme base un contenu en importations de la consommation privée de 41,38 % en 1981, on obtient

$C \xrightarrow{0,41} M.$

Ces différentes modifications des composants de la balance commerciale vont se répercuter dans une variation du solde de cette balance.



Les exportations nettes étant une composante du PNB au prix du marché, leur variation va entraîner une variation identique de celui-ci.
Et une variation de la consommation fera également varier ce PNB.

On représente ceci par

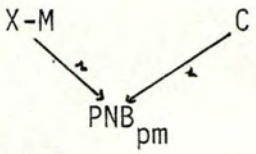
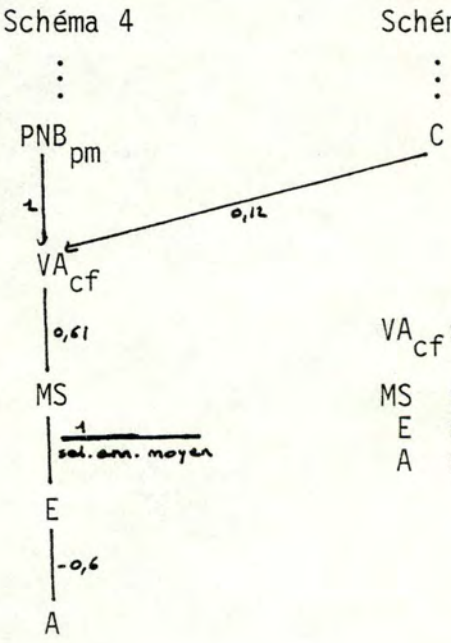


Schéma 5.



VA_{cf} : valeur ajoutée au coût des facteurs
MS : masse salariale brute
E : emploi
A : allocation de chômage

On a $PNB_{pm} \xrightarrow{1} VA_{cf}$

car PNB_{pm} - impôts directs + subventions = VA_{cf}

- les subventions ne varient pas dans le cadre de cette simulation
- on suppose que le taux moyen d'imposition indirecte est égal à 12 % des dépenses intérieures : 0,12 Δ C.

finalement $VA_{cf} = PNB_{pm} - 0{,}12\ \Delta C$.

La part de la masse salariale dans la valeur ajoutée pour l'ensemble de l'économie = 61,02 %, et donc $VA_{cf} \xrightarrow{0{,}61} MS$

ou ce pourcentage est obtenu comme suit :

$61{,}02 = (0{,}4909 \times 0{,}268) + (0{,}4909 \times 0{,}4974) + 0{,}2345$

où 0,4909 est le coéfficient de VA due au salaire, pour le secteur exposé et le secteur abrité privé.

Bien entendu, la valeur ajoutée dans le secteur abrité public est entièrement due aux salaires.

D'autre part, la modification de la masse salariale joue uniquement sur le nombre de personnes employées et non sur le taux de salaire.

Aussi cette influence se représente-t-elle comme suit :

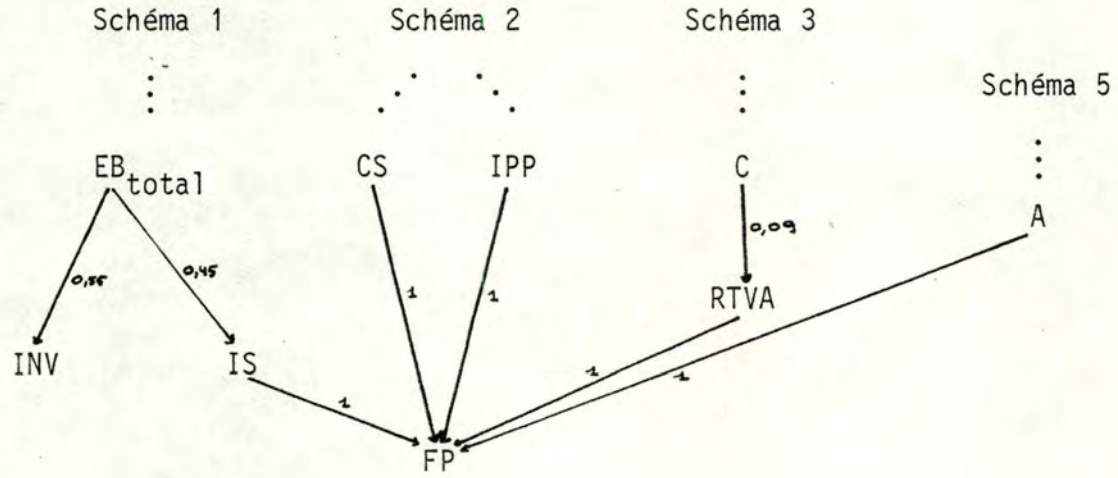
$MS \xrightarrow[\mu = \text{salaire annuel moyen}]{} E$ donc μ = 1 salaire annuel moyen après diminution de salaire.

La modification du nombre de personnes employées entraîne une modification similaire du nombre de personnes auxquelles est versée une allocation de chômage, pour autant que l'on suppose que la population active ne subit aucun changement.

On prend un taux d'allocation de chômage de 60 % du salaire brut, hors cotisations patronales.
Ainsi, en supposant que toute personne perdant son emploi a droit aux allocations de chômage, on peut établir l'influence suivante :

$$E \xrightarrow[-\ 0{,}60]{} A$$

Schéma 6.

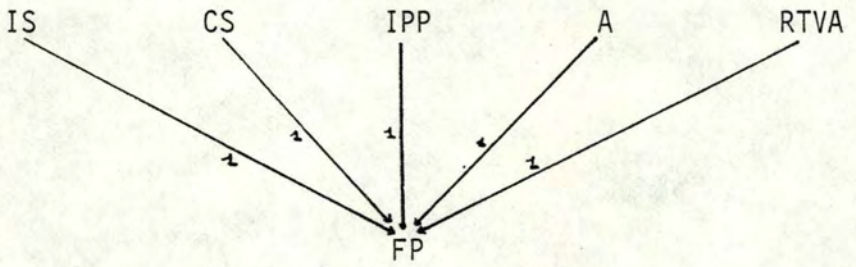


La croissance de l'excédent brut sera une incitation à de nouveaux investissements. Ceci est représenté par l'arc (EB, INV). Etant donné que les calculs relatifs aux investissements sont très complexes, le travail de Debatty-Ledouble n'en tient pas compte, on fera de même.
L'augmentation de l'excédent brut entraîne, d'autre part, une augmentation des recettes fiscales provenant de l'impôt de société : le montant de l'augmentation du bénéfice imposable est exactement égal au montant de l'augmentation de l'excédent brut. On calculera la variation de l'impôt des sociétés sur base d'un taux d'imposition de 45 %.

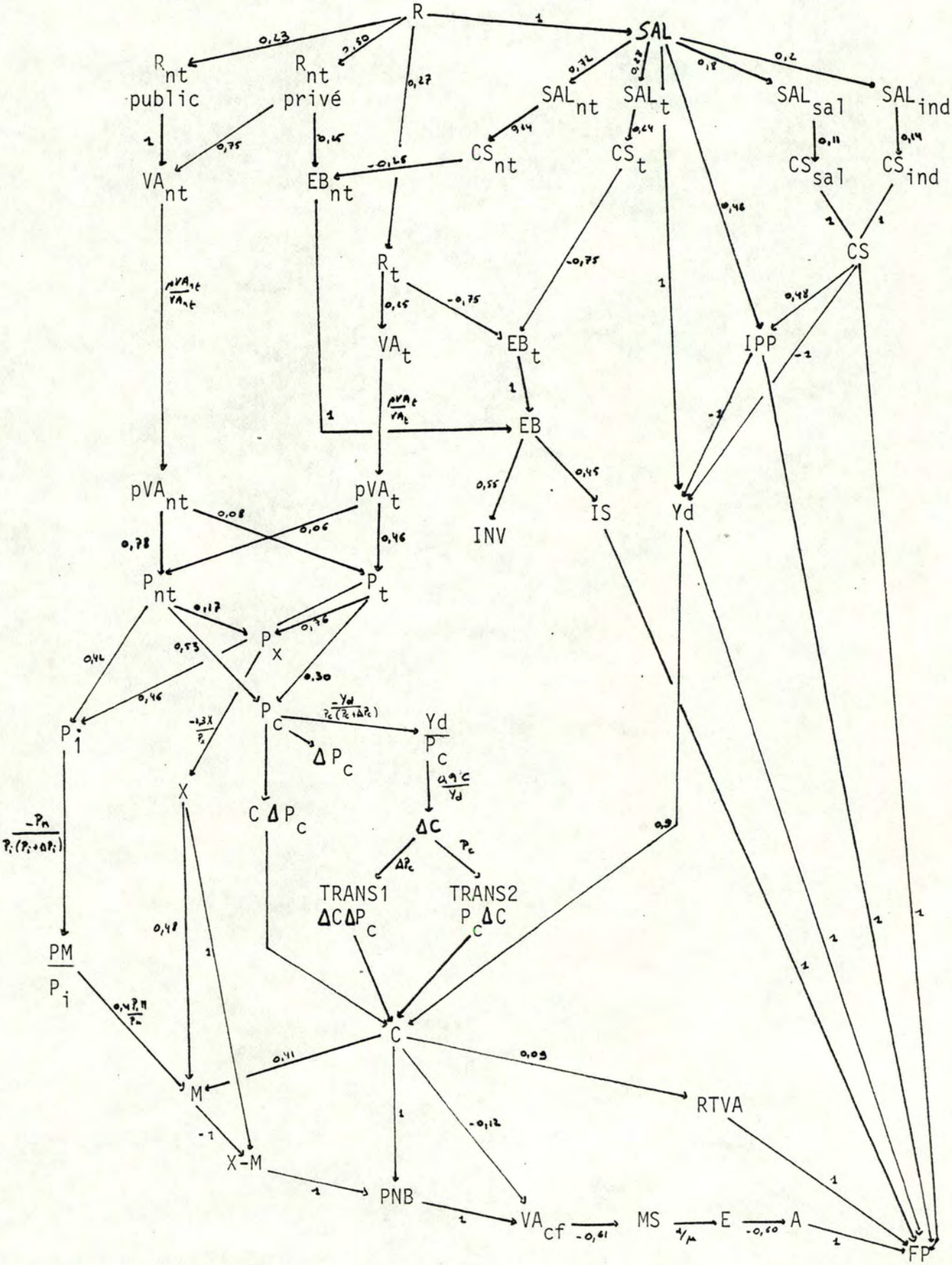
D'autre part, la réduction de la consommation privée des ménages réduit le montant des recettes de TVA.

$$C \xrightarrow{0,09} \text{RTVA, sur base d'un taux moyen de TVA de 9,51 \%.}$$

La variation de FP sera la somme des variations de IS, CS, IPP, RTVA, A.



Finalement, le schéma reconstitué donne ceci :



On prendra le jeu de valeurs des sommets suivant :

R : Revenu : 1.826.652 Mds (ici revenus salariaux)

$R_{nt\ privé}$: 908.667 Mds

$R_{nt\ public}$: 428.335 Mds

R_t : 489.649 Mds

SAL : salaires : 1.826.652 Mds

SAL_t : 489.649 Mds

SAL_{nt} : 1.337.003 Mds

SAL_{sal} : 0,8 x SAL = 1.461.312 Mds

SAL_{ind} : 0,2 x SAL = 365.330 Mds

* IPP : 394.042 Mds

+ CS : cotisation sociale : 154.585,3 Mds

+ CS sal : 115.401,3 Mds

+ CS ind : 39.184* Mds

Yd : revenu disponible : 2.825,6 Mds

VA_t : valeur ajoutée$_{revenu\ exposé}$: 489.649 Mds

VA_{nt} : valeur ajoutée$_{secteur\ abrité}$: 2.214.914 Mds

EB_t : excédent brut$_{secteur\ exposé}$: 459.674 Mds

EB_{nt} : excédent brut$_{secteur\ abrité}$: 877.912 Mds

EB : excédent brut$_{total}$: 1.337.586 Mds

C : consommation : 2.386,8 Mds

(*) X : Exportation : 2.062,3 Mds

M	: importations	: 2.476,2	Mds
* INV	: investissement	: 651.460	Mds
* IS	: impôts de société	: 14.974	Mds
* RTVA	: recettes de TVA	: 274.625	Mds
* FP	: finances publiques	: - 475,4	Mds
(*) PNB_{pm}	: produit national brut au prix du marché	: 3.266	Mds
E	: nombre d'employés	: 2.981	Millions
A	: chômeurs complets	: 521.370.	

* Economie belge en 82, MAE

(*) I F S

\+ Rapport général sur la sécurité sociale, Ministère Prév.Sociale.

Voici les resultats des differentes simulations effectuees lors de l'execution du programme.
Voici les resultats de votre ieme simulation.

Voici la liste des fleches constituant le scnena au moment de la simulation

origine	extremite	delai	coefficient
REV	RNTPU	0	2.3000E-01
REV	RNTPR	0	5.0000E-01
REV	RT	0	2.7000E-01
REV	SAL	0	1.0000E+00
RNTPU	VANT	0	1.0000E+00
RNTPR	VANT	0	7.5000E-01
RNTPR	EBNT	0	-2.5000E-01
SAL	SALNT	0	7.2000E-01
SAL	SALT	0	2.8000E-01
SAL	YD	0	1.0000E+00
SAL	SALSA	0	8.0000E-01
SAL	SALIN	0	2.0000E-01
SAL	IPP	0	4.8000E-01
VANT	PVANT	0	1.0000E-13
EBNT	EB	0	1.0000E+00
SALNT	CSNT	0	2.4000E-01
SALT	CST	0	2.4000E-01
SALSA	CSSAL	0	1.1000E-01
SALIN	CSIND	0	1.4000E-01
IPP	YD	0	-1.0000E+00
IPP	FP	0	1.0000E+00
RT	VAT	0	2.5000E-01
RT	EBT	0	-7.5000E-01
CSNT	EBNT	0	-2.5000E-01

CST	EBT	0	-7.5000E-01
CSSAL	CS	0	1.0000E+00
CSIND	CS	0	1.0000E+00
CS	IPP	0	-4.8000E-01
CS	YD	0	-1.0000E+00
CS	FP	0	1.0000E+00
VAT	PVAT	0	1.0000E-12
EBT	EB	0	1.0000E+00
YD	CONS	0	9.0000E-01
PVANT	PNT	0	7.8000E-01
PVANT	PT	0	8.0000E-02
PVAT	PNT	0	6.0000E-02
PVAT	PT	0	4.6000E-01
EB	IS	0	4.5000E-01
EB	INVES	0	5.5000E-01
IS	FP	0	1.0000E+00
PNT	PC	0	5.3000E-01
PNT	PX	0	1.7000E-01
PNT	PI	0	4.2000E-01
PT	PC	0	3.0000E-01
PT	PX	0	7.6000E-01
PT	PI	0	4.6000E-01
PC	DELPC	0	1.0000E+00
PC	TRA1	0	2.3860E+09
PC	YD/PC	0	8.0000E-01
PI	DELPI	0	1.0000E+00
PI	PM/PI	0	1.0000E+00
PM/PT	M	0	-4.0000E-01
M	X-M	0	-1.0000E+00
PX	X	0	-1.3000E+00

X	M	0	4.8000E-01
X	X-M	0	1.0000E+00
X-M	PNB	0	1.0000E+00
DELC	TRA3	0	1.0000E+00
DELC	TRA2	0	1.0000E+00
TRA1	CONS	0	1.0000E+00
TRA2	CONS	0	1.0000E+00
TRA3	CONS	0	1.0000E+00
CONS	PNB	0	1.0000E+00
CONS	VACF	0	1.2000E-01
CONS	RTVA	0	9.0000E-02
CONS	M	0	1.1000E-01
PNB	VACF	0	1.0000E+00
VACF	MS	0	6.1000E-01
MS	EMPL	0	5.2000E-01
EMPL	A	0	1.0000E+00
RTVA	FP	0	1.0000E+00
YD/PC	DELC	0	7.6000E-01
A	FP	0	1.0000E+00

voici la liste des centres de variation et leur valeur initiale associee, au moment de la simulation

l'unité est ici le million

REV	1.826652E+06
RNTPU	4.283350E+05
RNTPR	9.096670E+05
RT	4.956490E+05
SAL	1.826652E+06
VANT	2.214914E+06
EBNT	8.779120E+05
SALNT	4.896490E+05
SALT	1.333700E+06
YD	2.825600E+06
SALSA	1.461322E+06
SALIN	3.653300E+05
IPP	3.940420E+05
PVANT	0.000000E+00
EB	1.337586E+06
CSNT	1.175158E+05
CST	3.208807E+05
CSSAL	1.154010E+05
CSIND	3.918400E+04
FP	-4.754000E+05

```
VAT     ;       4.896490E+05
EBT     ;       4.596740E+05
CS      ;       2.985350E+05
PVAT    ;       0.000000E+00
CONS    ;       2.386800E+06
PNT     ;       0.000000E+00
PT      ;       0.000000E+00
IS      ;       1.497400E+04
INVES   ;       6.514600E+05
PC      ;       0.000000E+00
PX      ;       0.000000E+00
PI      ;       0.000000E+00
DELPC   ;       0.000000E+00
TRA1    ;       0.000000E+00
DELC    ;       0.000000E+00
YD/PC   ;       0.000000E+00
DELPI   ;       0.000000E+00
PM/PI   ;       0.000000E+00
M       ;       2.476200E+06
X-M     ;      -3.140000E+05
X       ;       2.062300E+06
PNB     ;       3.266000E+06
TRA2    ;       0.000000E+00
TRA3    ;       0.000000E+00
VACF    ;       0.000000E+00
RTVA    ;       2.746250E+05
MS      ;       1.826652E+06
EMPL    ;       2.981000E+00
A       ;       5.213700E-01
```

Voici les conditions initiales de votre simulation

cvd : REV

temps inparti : 0

impulsion de depart : -1.006300E+05

Voici le tableau recapitulatif des influences

voici les sommets influences apres un delai : 0

sommet	val. init	var.totale	nouv.valeur
REV	1.8267E+06	-1.0063E+05	1.7260E+06
RNTPU	1.2833E+05	-2.3145E+04	4.0519E+05
RNTPP	9.0867E+05	-5.0315E+04	8.5835E+05
RT	4.9565E+05	-2.7170E+04	4.6848E+05
SAL	1.8267E+06	-1.0063E+05	1.7260E+06
VANT	2.2149E+06	-6.0881E+04	2.1540E+06
EBNT	8.7791E+05	1.6926E+04	8.9484E+05
SALNT	4.8965E+05	-7.2454E+04	4.1720E+05

SALT	1.3337E+06	-2.8176E+04	1.3055E+06
YD	2.8256E+06	-4.6258E+04	2.7793E+06
SALSA	1.4613E+06	-8.0504E+04	1.3808E+06
SALIN	3.6533E+05	-2.0126E+04	3.4520E+05
IPP	3.9404E+05	-4.2699E+04	3.5134E+05
PVANT	0.0000E+00	-2.4352E-08	-2.4352E-08
EB	1.3376E+06	4.2375E+04	1.3800E+06
CSNT	1.1752E+05	-1.7389E+04	1.0013E+05
CST	3.2088E+05	-6.7623E+03	3.1412E+05
CSSAL	1.1540E+05	-8.8554E+03	1.0655E+05
CSIND	3.9184E+04	-2.8176E+03	3.6366E+04
FP	-1.7540E+05	-4.8435E+04	-5.2384E+05
VAT	4.8965E+05	-6.7925E+03	4.8286E+05
EBT	4.5967E+05	2.5449E+04	4.8512E+05
CS	2.9853E+05	-1.1673E+04	2.8686E+05
PVAT	0.0000E+00	-6.7925E-09	-6.7925E-09
CONS	2.3869E+06	-4.1660E+04	2.3451E+06
PNT	0.0000E+00	-1.9402E-08	-1.9402E-08
PT	0.0000E+00	-5.0728E-09	-5.0728E-09
IS	1.4974E+04	1.9069E+04	3.4043E+04
INVES	6.5146E+05	2.3306E+04	6.7477E+05
PC	0.0000E+00	-1.1805E-08	-1.1805E-08
PX	0.0000E+00	-7.1537E-09	-7.1537E-09
PI	0.0000E+00	-1.0483E-08	-1.0483E-08
DELPC	0.0000E+00	-1.1805E-08	-1.1805E-08
TRAT	0.0000E+00	-2.8167E+01	-2.8167E+01
DELC	0.0000E+00	-7.1775E-09	-7.1775E-09
YD/PC	0.0000E+00	-9.4441E-09	-9.4441E-09
DELPT	0.0000E+00	-1.0483E-08	-1.0483E-08
PM/PT	0.0000E+00	-1.0483E-08	-1.0483E-08

V	M	2.4762E+06	-1.7081E+04	2.4591E+06
X	X-M	-3.1400E+05	1.7081E+01	-2.9692E+05
X	X	2.0623E+06	9.2998E-09	2.0623E+06
V	PNB	3.2660E+06	-2.4579E+04	3.2414E+06
	IRA2	0.0000E+00	-7.1775E-09	-7.1775E-09
	IRA3	0.0000E+00	-7.1775E-09	-7.1775E-09
X	VACF	0.0000E+00	-2.9579E+04	-2.9579E+04
V	RIVA	2.7463E+05	-3.7494E+03	2.7088E+05
X	MS	1.8267E+06	-1.8043E+04	1.8086E+06
X	EMPL	2.9810E+00	-9.3823E+03	-9.3794E+03
X	A	5.2137E-01	-9.3823E+03	-9.3818E+03

N.B. : Les résultats obtenus ici sont conformes aux résultats obtenus par Debatty-Ledouble jusqu'à un certain point.

D'autres résultats apparaissent bizarres, ainsi les valeurs des variations de X, de X-M, MS, EMPL, VACF, A sont-elles totalement hirsutes.

Ceci est dû aux mauvaises valeurs posées sur certains arcs. L'ordinateur étant tombé en panne à plusieurs reprises fin mai, je n'ai pas eu l'occasion de modifier ces valeurs.

Les arcs à modifier sont :

$P_c \longrightarrow Y_{d/P_c}$	*dont le coéfficient est*	$-0,8$ *et non* $0,8$
$P_{m/P_i} \longrightarrow M$	*dont le coéfficient est*	$-0,4.\frac{M.P_i}{P_m}$ *et non* $0,4$
$P_x \longrightarrow X$	" " "	$-1,3\ \frac{X}{P_x}$ *et non* $-1,3$
delC $\longrightarrow$ *Tra3*	" " "	$DELP_c = -0,0118$ *et non* 1
MS $\longrightarrow$ *Empl.*	" " "	$\frac{1}{\text{rev.ann.moyen après modification}}$ *et non* $0,52$

Le schéma modifié, et les valeurs des résultats obtenus sur ce schéma seront présentés lors de la défense orale, si Sophie le veut bien ...

C) CONCLUSIONS de la simulation sur le modèle reflétant les influences sur l'économie belge d'une diminution des revenus.

On compare les résultats de cette simulation à ceux obtenus par Debatty et Ledouble.

Il faut remarquer tout d'abord que les valeurs des prix pVA_t, pVA_{nt} P_x, P_i, P_m, P_c n'ont pas été explicitées par Debatty-Ledouble dans leur rapport. Lors de la simulation, on les a posés à 0 de valeur initiale et à 1 lorsqu'ils interviennent dans le calcul d'un coéfficient.
Aussi obtient-on des résultats différents pour les valeurs finales de ces sommets et de leurs descendants.

Pour le calcul des influences où ces prix n'interviennent pas, on obtient des résultats similaires à ceux calculés dans leur rapport.

CONCLUSIONS

Ainsi, le programme présenté dans ce mémoire permet-il de construire des schémas économiques, d'effectuer des simulations sur ceux-ci grâce à un algorithme original. On aura, au préalable, vérifié par un procédé ne se trouvant pas dans la littérature connue, que le schéma ne comportait pas de circuit de délai nul, condition nécessaire à l'exécution de l'algorithme de récurrence.

Le programme autorise également de nombreuses et diverses modifications du schéma économique.

Finalement, ce programme permet de construire un fichier des sommets et de leurs valeurs, accordé au schéma économique sur lequel il travaille. Ceci permet d'effectuer des simulations sur un même schéma économique, pour des valeurs initiales différentes des sommets du graphe.

La plupart des difficultés énoncées lors de l'introduction ont pu être résolues. Ainsi le problème de la portabilité a-t-il partiellement trouvé une solution en isolant dans des composants spécifiques du programme les décisions de conception non conformes au PASCAL standard.
Seuls les problèmes inhérents au fait que les schémas économiques eux-mêmes n'ont aucune syntaxe précise de construction, ni de représentation, n'ont-ils pas trouvé de solution élégante.

On pourrait, par la suite, reprendre le programme et le modifier de façon à remédier à cela, par exemple en introduisant des composants dans la structure qui permettraient une allocation dynamique des valeurs aux coéfficients des arcs.

Une autre extension intéressante du programme serait de créer des composants qui garderaient automatiquement les valeurs des sommets après simulation. Ce qui permettrait de recommencer des simulations à partir de ces valeurs.

* * *

ANNEXE I

LE PROGRAMME

```
PROGRAM INTPODON(SCHEMA,FSOMMET,RESULTAT);

{* le premier fichier est celui qui represente le schema de travail,*}
{* ceci en debut comme en fin de programme *}

{* le deuxieme fichier est le fichier des sommets,*}
{* ceci en debut comme en fin de programme *}

{* le troisieme fichier est un fichier d'impression des resultats *}
{* c'est donc un fichier.txt *}

    const lindel = 25;

          {* tout delai devra etre inferieur ou egal a lindel *}

          blanc = '     ';
          nblettres = 5;

          {* nblettres represente le nombre de lettres que le nom *}
          {* d''un sommet du graphe peut comporter , au maximum *}

          liminfmes = -5;
          limsupmes = 5000000000;

          {* toute elasticite devra etre comprise entre liminfmes et limsupmes *}

          yaplus =  '$$$$$';
          help1 = 'help ';
          help2 = 'HELP ';

    type sommet = packed array [1..nblettres] of char;

          {* on considere que les sommets du graphe peuvent se *}
          {* representer par des mots de nblettres *}

          fleche = record
                       ext, ori : sommet;
                       del : integer;
                       taux : real
                   end;

          pteur =^arcnul;
          arcnul = record
                       alpha,omega : sommet;
                       numero : integer;
                       avant,arriere : pteur
                   end;

          ptsv =^somval;
          somval = record
                       som : sommet;
                       val : real;
                       valaj : real;
                       dvt,arri : ptsv;
                   end;

          cvval = record
                      cv : sommet;
                      valeur : real;
                  end;
```

```
var schema,schema2 : file of fleche;
    fsommet : file of cvval;
    resultat : file of char;
    tamb,buf : fleche;
    tampon : cvval;
    debut,fin,vv,teste : sommet;
    tps : integer;
    mes : real;
    nart,nbarcnul,indiccirc,nosim : integer;
    p,depart,arrivee,pcdt,pavancer,preculer,hier,demain: pteur;
    premier,arret : char;
    l,dpt,arvl,lavancer,lreculer,passe,futur : ptsv;
    prm : char;
    valinit : real;
    finfin,stopmemoire,arretrep,rep,message : char;
    pasXXX,pas : integer;
    stopsomval : char;
    reex,stopreex,arretansw,answ,stopdemande : char;
    valeurexiste : integer;
    texiste,impossible,correct : char;


procedure listsomval(n : integer);
{* ------------------------------- *}

{* cette procedure gere une liste des sommets *)
{* auxquels sont associes une valeur initiale *}

    begin
    case n of
    0: {* initialisation des pointeurs *}
       begin
           dpt:=nil;
           arvl:=nil;
           lavancer:=nil;lreculer:=nil;
           passe:= nil;futur:= nil;
           prm :='o';
       end;
    1: {* insertion d''un element en fin de liste *}
       begin
           if prm = 'o'
               then begin l:=arvl;
                          new(l);
                          dpt := l;
                          l^.som := vv;
                          if valeurexiste = 0 then
                              begin
                              write(tty,'quelle est la valeur initiale');
                              writeln(tty,' de ',vv,' ?');
                              readln(tty);
                              read(tty,valinit);
                              end;
                          l^.val := valinit;
                          l^.valaj:= 0;
                          l^.dvt := nil;
                          l^.arri := arvl;
                          prm:='n';
                          arvl := l;
```

```
                end
            else begin l:= arvl;
                       new(l);
                       arvl^.dvt := l;
                       l^.som := vv;
                       if valeurexiste=0 then
                           begin
                           write(tty,'quelle est la valeur initiale');
                           writeln(tty,' de ',vv , ' ?');
                           readln(tty);
                           read(tty,valinit);
                             end;
                           l^.val:= valinit;
                           l^.valaj := 0;
                           l^.dvt := nil;
                           l^.arri := arvl;
                           arvl:=l;
                   end;
    end;
2:  {* effacement du dernier element de la liste *}
    begin l:=arvl;
          arvl:=arvl^.arri;
          dispose(l);
          if arvl <> nil then arvl^.dvt := nil;
          l:=arvl;
    end;
3:  {* positionnement en debut de liste *}
    begin lavancer := dpt;
          l:= lavancer;
    end;
4:  {* positionnement en fin de liste *}
    begin lreculer := arvl;
          l:= lreculer;
    end;
5:  {* lire l''element suivant *}
    begin lavancer := lavancer^.dvt;
          l:=lavancer;
    end;
6:  {* lire l''element precedent *}
    begin lreculer := lreculer^.arri;
          l:=lreculer;
    end;
7:  {* effacer un element en milieu de liste *}
    begin if l^.arri <> nil then
        begin passe:= l^.arri;
              passe^.dvt := l^.dvt;
        end
        else dpt := dpt^.dvt;

          if l^.dvt <> nil then
              begin futur := l^.dvt;
                    futur^.arri:=l^.arri;
              end
          else arvl:=arvl^.arri;

              if (dpt = arvl) and (arvl = nil)
                  then prm:='o';

              dispose(l);
              l:=dpt;
```

```
        end;
    end;
    end;



procedure rechvalsom(XX : sommet);
{* ------------------------------- *}

{* Cette procedure recherche un sommet XX dans listsomval,*}
{* et envoie une valeur de pas nulle si le sommet ne s''y trouve pas *}

    var stoprech : char;


    begin
    listsomval(3);
    if l = nil then stoprech:='o' else stoprech:='n';
    pas := 0;
    while stoprech<>'o' do
        begin
        if l^.som = XX
            then begin
                 stoprech := 'o';
                 pas := 1;
                 end
            else if l^.dvt = nil then stoprech := 'o'
                 else l:= l^.dvt;
        end;
    end;


    procedure validsommet(qualite :integer);
{* ---------------------------------------- *}

{* cette procedure se charge de valider un sommet *}


      begin
      correct := 'o';
      case qualite of

      1 : {* validation d''un sommet en tant qu''origine d''un arc *}
            begin
            if debut = blanc then
                begin
                correct := 'n';
                writeln(tty,'votre origine n''a t''elle pas de nom ? ');
                end;
            end;

      2 : {* validation d''un sommet en tant qu''extremite d''un arc *}
            begin
            if fin = blanc then
                begin
                correct:= 'n';
                writeln(tty,'votre extremite n''a t''elle pas de nom ? ');
```

```
            end
        else if fin = debut
            then begin
                 correct:='n';
                 writeln(tty,'ce programme n''admet pas de boucle ');
                 end;
        end;

    3 : {* validation de l''existence d''un sommet *}
        begin
        recnvalsom(teste);
        if pas = 0 then begin
                        correct:= 'n';
                        if message = 'o'
                        then writeln(tty,'ce sommet n''existe pas ')
                        else message := 'o';
                        end;
        end;
    end;
    end;


    procedure validtemps(time : integer);
{* ------------------------------------ *}

{* cette procedure valide un delai donne *}

    begin
        correct:= 'o';
        if time < 0
        then begin
             correct:='n';
             writeln(tty,'ce programme n''admet pas de delai negatif ');
             end
        else if time > limdel
             then begin
                  correct:='n';
                  write(tty,'ce programme n''admet actuellement que ');
                  write(tty,'des delais inferieurs ou egaux a ',limdel);
                  writeln(tty);
                  end;
    end;


  procedure validmesure(coeff : real);
{* ---------------------------------- *}

{* cette procedure se charge de valider le coefficient donne a un arc *}

    begin
    correct := 'o';
    if (coeff < liminfmes) or (coeff > limsupmes)
        then begin correct:='n';
                  write(tty,'ce programme n''admet actuellement que');
                  write(tty,' des coefficients compris entre ');
                  writeln(tty,liminfmes,' et ',limsupmes);
             end;
    end;
```

```
procedure listarcnul(n : integer);
{* ------------------------------ *}

{* cette procedure gere une liste des arcs de delai nul *}


    begin
        case n of

        0: {* initialisation des pointeurs *}
           begin depart:=nil;
                 arrivee:=nil;
                 pavancer:=nil;preculer:=nil;
                 premier :='o'
           end;
        1: {* insertion d''un element en fin de liste *}
           begin if premier = 'o'
                 then begin p:=arrivee;
                            new(p);
                            depart := p;
                            p^.alpha:=debut;
                            p^.omega:=fin;
                            p^.numero := nbarcnul;
                            p^.avant:=nil;
                            p^.arriere:=arrivee;
                            premier := 'n';
                            arrivee := p;
                      end
                 else begin p:= arrivee;
                            new(p);
                            arrivee^.avant:= p;
                            p^.avant:=nil;
                            p^.alpha:=debut;
                            p^.omega:=fin;
                            p^.numero:=nbarcnul;
                            p^.arriere:=arrivee;
                            arrivee:=p;
                      end;
           end;

        2: {* effacement d''un element en fin de liste *}
           begin p := arrivee;
                 arrivee := arrivee^.arriere ;
                 dispose(p);
                 if arrivee <> nil then arrivee^.avant := nil ;
                 p:= arrivee;
           end;

        3: {* positionnement sur le premier element de la liste *}
           begin pavancer:= depart;p:= pavancer;end;

        4: {* positionnement sur le dernier element de la liste *}
           begin preculer := arrivee;p:=preculer;end;

        5: {* lecture de l''element suivant *}
           begin pavancer:=pavancer^.avant;p:=pavancer;end;
```

```
          6: (* lecture de l''element precedent *)
             begin preculer:=preculer^.arriere;p:=preculer;end;

          7: (* effacement d''un element quelconque de la liste *)
             begin if p^.arriere <> nil
                   then begin hier := p^.arriere;
                              hier^.avant:=p^.avant;
                        end
                   else depart:=depart^.avant;

                   if p^.avant <> nil
                   then begin demain:= p^.avant;
                              demain^.arriere:=p^.arriere;
                        end
                   else arrivee:= arrivee^.arriere;

                   if (depart = arrivee) and (arrivee = nil)
                       then premier:='o';

                   dispose(p);
                   p:=depart;
             end;
          end;
       end;


    procedure help(n:integer);
{*  ------------------------- *}

{* cette procedure permet d''envoyer des messages d''aide*}
{* a l''utilisateur lorsqu''il le desire, et que c''est possible *}

    begin
    case n of

    1 : (* explication sur l''introduction des donnees *)
        begin
        writeln(tty);
        write(tty,'Voudriez vous entrer les sommets de votre graphe,');
        writeln(tty,' origine par origine.');
        write(tty,'Et, pour chaque origine, il vous est demande de');
        write(tty,' fournir les extremites des fleches qui sont');
        writeln(tty,' issues de cette origine. ');
        writeln(tty);
        write(tty,'Si vous oubliez une fleche en cours de route,');
        write(tty,'veuillez l''introduire a l''aide des modifications');
        writeln(tty,' qui vous seront ulterieurement proposees. ');
        writeln(tty);
        write(tty,'C''est donc un nom d''au plus ',nblettres);
        writeln(tty,' lettres qui est attendu ici. ');
        write(tty,'Ce nom etant celui d''un sommet de votre graphe');
        writeln(tty,' origine d''une ou plusieurs fleches. ');
        write(tty,'Si tous les sommets-origines ont ete entres,');
        writeln(tty,' tapez  ',yaplus,'. ');
        writeln(tty);
        end;

    2 : (* explication sur l''introduction d''une extremite *)
        begin
        writeln(tty);
```

```
    write(tty,'Il vous est demande le nom d''un sommet extremite');
    writeln(tty,' d''une fleche dont l''origine est ',debut,'.');
    write(tty,'Ce nom devra comporter au plus ',nblettres);
    writeln(tty,' lettres.');
    writeln(tty);
    write(tty,'Si toutes les fleches issues du sommet ',debut);
    writeln(tty,' ont ete introduites,  tapez ',yaplus);
    writeln(tty);
    end;

3 : {* explications concernant le cvd *}
    begin
    writeln(tty);
    write(tty,'Le centre de variation de depart est le sommet');
    write(tty,' du graphe qui subit une modification dont');
    writeln(tty,' l''ampleur vous sera demandee. ');
    write(tty,'Le programme est fait pour calculer les effets de');
    write(tty,' cette modification sur les autres sommets du graphe. ');
    writeln(tty);
    writeln(tty);
    write(tty,'C''est donc le nom de ce sommet qui vous est demande. ');
    writeln(tty);
    writeln(tty);
    end;

4 : {* explications pour l''ajout d''une fleche *}
    begin
    writeln(tty);
    write(tty,'Il vous est demande le nom du sommet origine de la');
    writeln(tty,' fleche que vous voulez ajouter  a votre schema. ');
    writeln(tty);
    write(tty,'Ce nom peut comporter ',nblettres,' lettres');
    writeln(tty,' au plus .');
    writeln(tty);
    write(tty,'Ce sommet peut ne pas exister deja, dans ce cas,');
    write(tty,' il vous sera demande de donner une valeur a ce');
    writeln(tty,' sommet. ');
    writeln(tty);
    end;

5 : {* explications pour l''ajout d''une fleche *}
    begin
    write(tty,'Il vous est demande le nom du sommet extremite de ');
    writeln(tty,'la fleche que vous voulez ajouter a votre schema. ');
    writeln(tty);
    writeln(tty,'Ce nom peut comporter ',nblettres,' lettres au plus .');
    writeln(tty);
    write(tty,'Ce sommet peut ne pas exister deja, dans ce cas, il');
    writeln(tty,' vous sera demande de donner une valeur a ce sommet. ');
    writeln(tty);
    end;

6 : {* explications pour le retrait d''une fleche *}
    begin
    writeln(tty);
    write(tty,'Il vous est demande le nom du sommet origine de la');
    writeln(tty,' fleche que vous voulez retirer de votre schema. ');
    write(tty,'Si, par hasard, ce sommet n''existait pas,');
    write(tty,'le programme vous le signalera, et vous sortirez de');
    writeln(tty,' la modification courante. ');
```

```
    writeln(tty);
    end;

7 : {* explications pour le retrait d''une fleche *}
    begin
    writeln(tty);
    write(tty,'Il vous est demande le nom du sommet extremite de la');
    writeln(tty,' fleche que vous voulez retirer de votre schema. ');
    write(tty,'Si, par hasard, ce sommet n''existait pas,');
    write(tty,'le programme vous le signalera, et vous sortirez de');
    writeln(tty,' la modification en cours. ');
    writeln(tty);
    end;

8 : {* explications pour la modification d''une fleche *}
    begin
    writeln(tty);
    write(tty,'Il vous est demande le nom du sommet origine de la');
    writeln(tty,' fleche dont vous voulez modifier les valeurs. ');
    write(tty,'Si, par hasard, ce sommet n''existait pas,');
    write(tty,' le programme vous le signalera et vous sortirez de');
    writeln(tty,' la modification en cours.');
    writeln(tty);
    end;

9 : {* explications pour la modification d''une fleche *}
    begin
    writeln(tty);
    write(tty,'Il vous est demande le nom du sommet extremite de la');
    writeln(tty,' fleche dont vous voulez modifier les valeurs. ');
    write(tty,'Si, par hasard, ce sommet n''existait pas,');
    write(tty,'le programme vous le signalera et vous sortirez de');
    writeln(tty,' la modification en cours.');
    writeln(tty);
    end;

10: {* explications sur les possibilites de modification du schema donne *}
    begin
    writeln(tty);
    write(tty,'Il vous est possible d''ajouter une nouvelle');
    write(tty,' fleche dans le schema, d''en retirer une,');
    write(tty,' ou de modifier les valeurs attachees a l''une');
    writeln(tty,' d''entre elles.');
    write(tty,'Il vous est egalement possible de retirer un');
    write(tty,' sommet du graphe ou de modifier la valeur qui');
    writeln(tty,' lui correspond. ');
    writeln(tty);
    write(tty,'Si une de ces possibilites vous interesse,');
    writeln(tty,' repondez o pour oui.');
    writeln(tty);
    write(tty,'Si vous n''avez nullement l''intention de');
    writeln(tty,' modifier votre schema, repondez n pour non. ');
    write(tty,'Le programme vous demandera alors si vous desirez');
    writeln(tty,' executer le programme avec le schema entre. ');
    writeln(tty);
    end;

11: {* explication de l''execution du programme avec le meme schema *}
    begin
    writeln(tty);
```

```
        write(tty,'Si vous desirez recommencer la simulation avec');
        write(tty,' un centre de variation de depart different,');
        write(tty,'ou pour un temps imparti different ou encore ');
        write(tty,' avec une autre imulsion de depart, repondez o ');
        writeln(tty,' pour oui. ');
        writeln(tty);
        write(tty,'Si cela ne vous interesse pas , repondez n pou');
        writeln(tty,'r non.');
        write(tty,'Le programme vous presentera alors le schema tel');
        write(tty,'que vous l''avez dernierement entre, et vous');
        writeln(tty,' proposera de le modifier. ');
        writeln(tty);
        end;

12:  {* explications sur l''execution *}
     begin
     writeln(tty);
     write(tty,'Si vous desirez executer une simulation sur le');
     write(tty,' schema que vous avez entre en dernier lieu,');
     writeln(tty,' repondez o pour oui.');
     writeln(tty);
     write(tty,'Si vous ne voulez pas simuler, repondez n pour non.');
     writeln(tty);
     writeln(tty);
     writeln(tty,'Vous sortez alors definitivement du programme !!!');
     writeln(tty);
     end;

13:  {*explications sur le retrait d''un sommet *}
     begin
     writeln(tty);
     write(tty,'Il vous est demande le nom du sommet que vous');
     writeln(tty,' desirez retirer de votre graphe.');
     write(tty,'Si, par hasard, ce sommet n''existait pas,');
     write(tty,'le programme vous le signalera, et vous sortirez de');
     writeln(tty,' la modification en cours.');
     writeln(tty);
     write(tty,'Retirez un sommet du graphe revient egalement a ');
     write(tty,' retirer toutes les fleches ');
     writeln(tty,'qui lui sont adjacentes. ');
     writeln(tty);
     end;

14:  {* explications sur une modification de valeur d''un sommet *}
     begin
     writeln(tty);
     write(tty,'Il vous est demande le nom du sommet dont vous');
     writeln(tty,' voulez modifier la valeur,');
     write(tty,'Si, par hasard, ce sommet n''existait pas, le');
     write(tty,' programme vous le signalerait,');
     writeln(tty,'et vous seriez sorti de la modification en cours.');
     writeln(tty);
     write(tty,'Si le sommet existe, le programme vous presentera ');
     write(tty,' son ancienne valeur ');
     writeln(tty,'et vous demandera sa nouvelle valeur. ');
     writeln(tty);
     end;

15:  {* explications sur l''affichage des resultats de modifications *}
     begin
```

```
        writeln(tty);
        write(tty,'En repondant o pour oui, un tableau representant ');
        write(tty,' les arcs de votre schema sera affiche ');
        write(tty,',ainsi qu''une liste des valeurs des sommets de ');
        writeln(tty,' votre schema.');
        write(tty,'Si vous repondez n pour non, le programme vous');
        write(tty,'offre alors la possibilite de nouvelles modifications. ');
        writeln(tty);
        end;

    16: (* explications sur l'impression des resultats *)
        begin
        write(tty,'En repondant o pour oui,un tableau representant par');
        write(tty,' delai, et par sommet les influences ');
        write(tty,' declenchees par la variation du centre ');
        write(tty,' de depart sera inscrit dans le fichier resultat.');
        write(tty,' Si vous desirez un listing de ce tableau, vous');
        write(tty,' n''aurez plus qu''a faire imprimer ce fichier');
        writeln(tty,' resultat');
        writeln(tty);
        write(tty,'Si vous ne desirez pas garder de trace de votre');
        write(tty,' derniere simulation');
        writeln(tty,'repondez n pour non ');
        writeln(tty);
        end;

    17: (* explications sur la donnees des operations dans le detail *)
        begin
        writeln(tty);
        write(tty,'Si vous repondez o pour oui, vous verrez apparaitre');
        write(tty,' une liste reprenant pour chaque delai different,');
        write(tty,' la suite des sommets touches par une variation ');
        write(tty,' du centre de depart, ainsi que la mesure de ces');
        write(tty,' influences. Un meme sommet sera repris, pour un');
        write(tty,' un delai donne, autant de fois qu''il ne subit');
        writeln(tty,' d''influences differentes ');
        writeln(tty);
        write(tty,'Si vous ne desirez pas connaitre le detail des ');
        write(tty,'operations qui ont permis d''etablir le tableau ');
        writeln(tty,'recapitulatif precedent, alors tapez n pour non ');
        writeln(tty);
        end;

    end;
    end;


    procedure introduction(x:char);
(*  ------------------------------ *)

{* cette procedure permet d''entrer les arcs *}
{* et je les ranger dans le fichier de sortie *}


        var stop,stopdebut,stopfin,stoptps,stopmes : char;

{* corps de la procedure introduction *}
{* ********************************** *}
```

```
          begin
          stopdebut := 'n';
          while stopdebut<>'o' do
           begin
           writeln(tty,'donnez un sommet origine d''une ou plusieurs fleches');
           writeln(tty,'                          *******');
           write(tty,'tapez ',yaplus,' si tous les sommets du schema ont');
           writeln(tty,' ete epuises');

           readln(tty);
           read (tty, debut);
           if (debut = help1)or(debut = help2) then help(1)
              else begin
                   validsommet(1);
                   if correct = 'o' then stopdebut :='o';
                   end;
           end;

{* on arrete la procedure lorsque l''origine donnee vaut yaplus *}
{* ------------------------------------------------------------ *}
          if debut = yaplus then begin arret:= 'o';
                                 stop :='o'
                                   end
              else begin stop := 'n';
                         rechvalsom(debut);
                         if pas = 0 then begin vv := debut;
                                         listsomval(1);
                                           end;

                         {* on place debut dans la liste des sommets,*}
                         {* pour autant qu''il ne s''y trouve pas deja *}

                   end;

              while stop = 'n' do
                  begin stopfin := 'n';
                        while stopfin<>'o' do
                        begin
                        write(tty,'donnez un sommet extremite d''une fleche');
                        writeln(tty,' dont l''origine est :',debut:nblettres);
                        writeln(tty,'                      *********');
                       writeln(tty,' tapez ',yaplus,' quand il n''y en a plus.');
                        readln (tty);
                        read (tty,fin);
                        if (fin = help1) or (fin = help2) then help(2)
                            else begin
                                 validsommet(2);
                                 if correct = 'o' then stopfin :='o';
                                 end;
                        end;
             {* si fin = yaplus on ne demande pas le delai et l''elasticite *}
             {* ----------------------------------------------------------- *}

                        if fin <> yaplus then
                            begin rechvalsom(fin);
                                  if pas = 0 then
                                   begin vv := fin;
                                         listsomval(1);
                                   end;
```

```
                        {* on place fin dans la liste des sommets*}
                        {* pour autant qu''il ne s''y trouve pas deja *}

                            stoptps:='n';
                            while stoptps<>'o' do
                            begin
                          write(tty,'donnez le delai del l''arc');
                          write(tty,' d''origine :',debut:nblettres);
                        writeln(tty,' et d''extremite :',fin:nblettres);
                          writeln (tty,'                *****');
                          readln(tty);
                          read (tty, tps);

 {* le delai donne doit etre compris entre 0 et limdel *}

                          validtemps(tps);
                          if correct = 'o' then stoptps:='o';
                            end;

                            stopmes:='n';
                            while stopmes<>'o' do
                              begin
                          write (tty, 'donnez le coefficent de l''arc');
                           write(tty,' d''origine :',debut:nblettres);
                          write(tty,' et d''extremite :',fin:nblettres);
                           writeln(tty);
                           writeln (tty, '           **********');
                           readln(tty);
                           read (tty, mes);

{* l''elasticite doit etre comprises entre liminfmes et limsupmes *}

                           validmesure(mes);
                           if correct = 'o' then stopmes:='o';
                              end;

                           tamp.ori := debut;
                           tamp.ext := fin;
                           tamp.del := tps;
                           tamp.taux := mes;
                           schema^ := tamp;
                           put (schema);
                           nart:=nart + 1;

{* nart calcule le nombre d''articles dans le fichier *}
{* ce qui equivaut au nombre d''arcs dans le schema *}

                           if tps = 0 then begin
                                             nbarcnul :=nbarcnul +1;

{* nbarcnul calcule le nombre d''arcs nuls dans le schema *}

                                             listarcnul(1);
                                           end
                           end
                                     else stop := 'o';
                                          end
                end;
```

```
   procedure affichage;
{* ---------------------- *}

{* cette procedure affiche a l''ecran les arcs du schema *}
{* avec leurs valeurs, ainsi que la liste des sommets du schema *}
{* et leurs valeurs initiales *}

 var stop1,stoplist : char;


 begin
 reset(schema);
 if eof(schema) then stop1:='o' else stop1:='n';
 writeln(tty,'voici la liste des fleches du schema ');
 writeln(tty,'------------------------------------ ');
 writeln(tty);
 writeln(tty,'  origine     extremite      delai      coefficient ');
 writeln(tty,'-------------!------------!-----------!------------');
if stop1 = 'o' then writeln(tty,'ce schema ne contient aucune fleche');
 while stop1<>'o' do
 begin
 tamp:=schema^;
writeln(tty,tamp.ori:12,'!',tamp.ext:12,'!',tamp.del:12,'!',tamp.taux:12);
writeln(tty,'-------------!------------!-----------!------------');
 get(schema);
 if eof(schema) then stop1:='o';
 end;
 close(schema);


 listsomval(3);
 if l= nil then stoplist:='o'
     else begin
          stoplist:='n';
 write(tty,'voici la liste des centres de variation et leur valeur');
 writeln(tty,' initiale associee');
 write(tty,'-------------------------------------------------------');
 writeln(tty,'--------------------');
 while stoplist<>'o' do
     begin
     writeln(tty,l^.som,' : ',l^.val);
     writeln(tty);
     if l^.dvt <> nil then l:=l^.dvt
                      else stoplist:='o';
     end;
          end;
 end;


   procedure demandemodification;
{* ---------------------------- *}

{* cette procedure appelle une procedure de modification du schema *}          {* si l''utilisateur

     var modifie,affich,stopaffich:char;


   procedure modification;
{* ---------------------- *}
```

```
{* Cette procedure propose un menu des differentes modifications *)
{* possibles sur le schema *}

        var n : integer;
            arretmenu : char;


          procedure recopiedau;
        {* ---------------------  *}

        {* cette procedure recopie le contenu du fichier schema2 *}
        {*d ans le fichier schema *}

        var stoplect2: char;


        begin
        reset(schema2,'SCHEMA2.SEQ');
        rewrite(schema);
        if eof(schema2) then stoplect2:='o' else stoplect2:='n';
        while stoplect2 <>'o' do
            begin
            tamp := schema2^;
            schema^ := tamp;
            put(schema);
            get(schema2);
            if eof(schema2) then stoplect2 := 'o';
            end;
        close(schema);
        close(schema2);
        end;


  procedure ajfleche;
{ ------------------  *)

{* cette procedure place une nouvelle fleche dans le fichier schema, *}
{* a la suite de celles de meme origine.*}

    var inutile,stoplects,stoporiaj,stopextaj : char;
        stopdelaj,stoptauxaj : char;
        tauxaj : real;
        serie,delaj,mis : integer;


    begin
    stoporiaj:='n';
    while stoporiaj<>'o' do
        begin
        writeln(tty,'quelle est l''origine de la nouvelle fleche ?');
        readln(tty);
        read(tty,debut);
        if (debut = help1)or(debut = help2) then help(4)
            else begin
                validsommet(1);
                if correct = 'o' then stoporiaj:='o';
                end;
```

```
    end;
rechvalsom(debut);
if pas = 0 then begin vv := debut;
                listsomval(1);
                end;

{* si ce debut est un nouveau sommet, on doit demander sa valeur *}
{* et le placer dans listsomval *}

stopextaj:='n';
while stopextaj<>'o' do
    begin
    writeln(tty,'quelle est l''extremite de la nouvelle fleche ?');
    readln(tty);
    read(tty,fin);
    if (fin = help1)or(fin = help2) then help(5)
        else begin
             validsommet(2);
             if correct = 'o' then stopextaj:='o';
             end;
    end;
rechvalsom(fin);
if pas = 0 then begin vv:= fin;
                listsomval(1);
                end;


{* si cette extremite est un nouveau sommet, *}
{* on doit demander sa valeur initiale et le placer dans listsomval *}

reset(schema);
rewrite(schema2,'schema2.seq');
if eof(schema) then stoplects :='o'
    else stoplects :='n';
inutile :='n';

{* inutile indique s''il est utile ou non de recopier schema2 *}
{* dans schema1. Si, par exemple, la fleche que l''on veut rajouter *}
{* existe deja, inutile prend la valeur o pour oui.*}
{* mais, a priori, on suppose que ce recopiage sera utile *}

serie := 0;
mis:=0;

{* mis indique si la fleche a ete placee dans le fichier ou pas *}

while stoplects <>'o' do
    begin
    tamp := schema^;
    if tamp.ori = debut
    then begin
         if tamp.ext = fin
         then begin
              stoplects := 'o';
              inutile := 'o';
              write(tty,'cette fleche existe deja , vous pouvez ');
              write(tty,'modifier les valeurs qui lui sont attachees');
              writeln(tty,' en executant l''operation 4');
              mis:=1;
              end
```

```
                else begin
                     serie := 1;
                     schema2^:= tamp;
                     put(schema2);
                     end;
                end
           else begin
                if serie = 0
                   then begin
                        schema2^ := tamp;
                        put(schema2);
                        end
                   else begin
                        stopdelaj:='n';
                        while stopdelaj<>'o' do
                              begin
                          write(tty,'quel est le delai de l''arc d''origi');
                          writeln(tty,'ne ',debut,' et d''extremite ',fin,' ?');
                          readln(tty);
                          read(tty,delaj);
                          validtemps(delaj);
                          if correct = 'o' then stopdelaj:='o';
                              end;
                          stoptauxaj:='n';
                          while stoptauxaj<>'o' do
                              begin
                        write(tty,'quelle est le coefficient de l''arc d''ori');
                        writeln(tty,'gine ',debut,' et d''extremite ',fin,' ?');
                          readln(tty);
                          read(tty,tauxaj);
                          validmesure(tauxaj);
                          if correct = 'o' then stoptauxaj:='o';
                              end;
                          buf.ori:=debut;
                          buf.ext:=fin;
                          buf.del:=delaj;
                          buf.taux:=tauxaj;
                          schema2^:=buf;
                          put(schema2);
                          nart:=nart+1;
                          mis:=1;

{* si on a pu placer la nouvelle fleche dans le fichier,mis prend la valeur 1*}
{* et nart s''incremente d''une unite *}

                          if delaj = 0 then begin
                                            nbarcnul:=nbarcnul+1;
                                            listarcnul(i);
                                            end;

                          {* si le delai de la nouvelle flechal est nul *}
                          {* alors nbarcnul s''incremente d''une unite *}
                          {* et on place la nouvelle fleche dans listarcnul *}

                          schema2^:=tamp;
                          put(schema2);
                          serie:=0;
                          end;
                end;
           get(schema);
```

```
            if eof(schema) then stoplects:='o';
            end;
        if mis = 0 then begin

{* si on arrive en fin du fichier sans avoir pu placer la nouvelle fleche *}
{* alors celle ci doit se placer en fin de fichier *}

                        stopdelaj:='n';
                        while stopdelaj<>'o' do
                            begin
                write(tty,'quel est le delai de l'' arc d''origine ');
                writeln(tty,debut,' et d''extremite ',fin,' ?');
                readln(tty);
                read(tty,delaj);
                validtemps(delaj);
                if correct = 'o' then stopdelaj:='o';
                        end;
                        stoptauxaj:='n';
                        while stoptauxaj<>'o' do
                            begin
                write(tty,'quelle est le coefficient de l''arc d''origine');
                writeln(tty,' ',debut,' et d'' extremite ',fin,' ?');
                readln(tty);
                read(tty,tauxaj);
                validmesure(tauxaj);
                if correct = 'o' then stoptauxaj:='o';
                        end;
                        buf.ori:=debut;
                        buf.ext:=fin;
                        buf.del:=delaj;
                        buf.taux:=tauxaj;
                        schema2^:=buf;
                        put(schema2);
                        nart:=nart+1;
                        if delaj=0 then begin
                                        nbarcnul:=nbarcnul+1;
                                        listarcnul(1);
                                        end;
                    end;
    close(schema);
    close(schema2);
    if inutile='n' then recopiedau;
    end;


    procedure retfleche;
{* ---------------------- *}

{* cette procedure permet de retirer une fleche du graphe! *}
{* mais ne retire pas pour autant les sommets origine *}
{* et extremite de cette fleche *}

var debutret,finret : sommet;
    stoplects,stoprecharc,stopdebutret,stopfinret:char;
    passage : integer;


begin
```

```
stopdebutret:='n';
while stopdebutret <>'o' do
    begin
    write(tty,'quelle est l''origine de la fleche que vous');
    writeln(tty,' voulez retirer ?');
    readln(tty);
    read(tty,debutret);
    if (debutret = help1)or(debutret = help2) then help(6)
       else stopdebutret:='o';
    end;


{* on verifie que ce debut est bien un sommet existant *}
{* sinon on ne peut certes pas retirer une fleche qui  *}
{* aurait pour origine un sommet inexistant ! *}

teste := debutret;
validsommet(3);
if correct = 'o'
 then begin
      stopfinret:='n';
      while stopfinret<>'o' do
         begin
      write(tty,'quelle est l''extremite de la fleche quel vous ');
      writeln(tty,' voulez retirer ?');
      readln(tty);
      read(tty,finret);
      if (finret = help1)or(finret = help2) then help(7)
          else stopfinret:='o';
          end;

      {* on verifie que cette extremite est bien un sommet existant *}
      {* sinon on ne peut pas retirer une fleche dont l''extremite  *}
      {* est un sommet inexistant ! *}

      teste:=finret;
      validsommet(3);
      if correct = 'o'
       then begin
            reset(schema);
            rewrite(schema2,'schema2.seq');
            if eof(schema) then
               begin
      write(tty,'le graphe ne comporte aucune fleche, on ne peut');
      writeln(tty,' donc pas en retirer');
      stoplects:='o';
               end

   {* il se peut que les sommets origine et extremite de la fleche *}
   {* a retirer existent mais que le fichier ne comporte aucun article *}
   {* on ne sait donc pas retirer d''article, on en informe l''utilisateur *}

            else begin
                 stoplects:='n';
                 passage:=0;
                 while stoplects<>'o' do
                     begin
                     tamp:=schema^;
                     if (tamp.ori=debutret)and(tamp.ext=finret)
                     then begin
```

```
                                passage:=1;

{* si la fleche que l''on retire avait un delai nul, il faut egalement *}
{* la retirer de listarcnul *}

                                if tamp.del =0
                                then begin
                                     nbarcnul:=nbarcnul-1;
                                     listarcnul(3);
                                     if p= nil then stoprecharc:='o'
                                        else stoprecharc:='n';
                                     while stoprecharc<>'o' do
                                        begin
                            if (p^.alpha = debutret) and (p^.omega = finret)
                              then begin listarcnul(7);
                                         stoprecharc:='o';
                                   end
                              else if p^.avant <> nil
                                   then p:= p^.avant
                                   else stoprecharc:='o';
                                        end;
                                     end;
                                end
                             else begin
                                  schema2^:=tamp;
                                  put(schema2);
                                  end;
                             get(schema);
                             if eof(schema) then stoplects:='o';
                             end;
                        close(schema);
                        close(schema2);
                        if passage =0
                             then writeln(tty,'cette fleche n''existe pas')
                                 else begin nart:=nart-1;
                                            recopiedau;
                                      end;
                        end;
                end;
        end;
    end;



    procedure retsom;
{* ------------------ *}

{* cette procedure permet de retirer un sommet de la liste des sommets *}
{* et de retirer du fichier schema,tous les arcs adjacents a ce sommet *}

    var somret : sommet;
        stopsomret,stoplect,stoprecharc:char;



    begin
    reset(schema);
    rewrite(schema2,'schema2.seq');
```

```
stopsomret:='n';
while stopsomret<>'o' do
     begin
     writeln(tty,'quel est le sommet que vous desirez retirer ?');
     readln(tty);
     read(tty,somret);
     if (somret = help1)or(somret = help2) then help(13)
          else stopsomret:='o';
     end;

{* on verifie que ce sommet existe sinon on a termine *}

teste := somret;
validsommet(3);
if correct = 'n'
then begin
     close (schema);
     close(schema2);
     end
else begin
     if eof(schema) then stoplect:='o' else stoplect :='n';

     {* si le fichier est vide, on a rien a retirer dedans *}

     while stoplect<>'o' do
          begin
          temp := schema^;

          {* les articles dont l''origine et l''extremite sont *}
          {* differentes du sommet a retirer, ne sont pas changes *}

          if(temp.ori<>somret) and (temp.ext<>somret)
               then begin schema2^:=temp;
                          put(schema2);
                    end

               {* les autres articles sont retires *}

               else begin
                    nart:=nart-1;


                    {* si la fleche retiree avait un delai nul *}
                    {* il faut egalement la retirer de listarcnul *}

                    if temp.del=0
                       then begin
                            noarcnul:=noarcnul-1;
                            listarcnul(3);
                            if p = nil then stoprecharc:='o'
                               else stoprecharc:='n';
                            while stoprecharc <> 'o' do
                               begin
                     if (p^.alpha = temp.ori) and (p^.omega = temp.ext)
                     then begin listarcnul(7);
                                stoprecharc:='o';
                          end
                     else if p^.avant <>nil then p:= p^.avant
                          else stoprecharc:='o';
                                end;
```

```
                                end;
                        end;
                    get (schema);
                    if eof(schema)then stoplect :='o';
                end;
            close(schema);
            close(schema2);
            recopiedau;
            listsomval(7);

            {* on efface le sommet de la liste des sommets. *}
            {* On est sur de se trouver sur le bon element  *}
            {* grace a rechvalsom qui s''est positionne a cet endroit *}

            end;
    end;



procedure modfleche;
{* -------------------- *}

{* cette procedure permet de modifier les valeurs associees a un arc *}
{* a savoir le delai et l''elasticite *}

    var arretlect,on,don,stopon,stopdon,stoprecharc : char;
        delmod,marque : integer;
        tauxmod : real;
        stopmodfeb,stopmodfin,stopdelmod,stoptauxmod : char;
        modific : char;


    begin
    marque := 0;

{* marque reste a 0 pour indiquer qu''il est necessaire de recopier *}
{* le schema2 dans schema1 *}

    modific:='n';
    stopmodfeb:='n';
    while stopmodfeb<>'o' do
     begin
     writeln(tty,'quelle est l''origine de la fleche que vous voulez modifier ?');
     readln(tty);
     read(tty,debut);
     if (debut = help1)or(debut = help2) then help(8)
         else stopmodfeb:='o';
     end;

{* la fleche a modifier n''existe surement pas si son origine n''existe pas *}

    teste := debut;
    validsommet(3);
    if correct = 'o'
    then begin
         stopmodfin:='n';
         while stopmodfin<>'o' do
             begin
             write(tty,'quelle est l''extremite de la fleche que vous ');
```

```
            writeln(tty,' voulez modifier ?');
            readln(tty);
            read(tty,fin);
            if (fin = help1)or(fin = help2) then help(9)
               else stopmodfin:='o'
            end;

        {* la fleche a modifier n''existe pas si son extremite n''existe pas *}

         reste:= fin;
         validsommet(3);
         if correct = 'o' then

            begin
            reset(schema);
            rewrite(schema2,'schema2.seq');
            if eof(schema)
               then begin arretlect:='o';
                          marque:=1;

{* si le fichier ne comprend aucun article, alors il est inutile *}
{* de recopier schema2 dans schema1,de plus, on averti l''utilisateur *}

                          write(tty,'ce schema ne contient aucune fleche,');
                          writeln(tty,'on ne peut donc pas en modifier une');
                    end
               else arretlect := 'n';
               while arretlect <>'o' do
                begin
                tamp := schema^;
                delmod := tamp.del;
                tauxmod := tamp.taux;
                if (tamp.ori = debut) and (tamp.ext = fin)
                    then begin
                         modific := 'o';
                         stopon:='n';
                         while stopon<>'o' do
                             begin
                             writeln(tty,'voulez-vous modifier le delai ?');
                             readln(tty);
                             read(tty,on);
                             if (on = 'o') or (on = 'O')
                             then begin
                                  stopon:='o';
                                  stopdelmod:='n';
                                  while stopdelmod<>'o' do
                                    begin
            writeln(tty,'l''ancienne valeur du delai etait : ',tamp.del);
            writeln(tty,'donnez la nouvelle valeur du delai ');
            readln(tty);
            read(tty,delmod);
            validtemps(delmod);
            if correct = 'o' then stopdelmod:='o';
                                    end;
                                  if (tamp.del = 0) and (delmod<>tamp.del)


{* si l''ancien delai valait 0 et que le nouveau delai ne vaut plus 0 *}
{* alors il faut effacer cette fleche de listarcnul *}
                                     then begin
```

```
                                        nbarcnul:=nbarcnul-1;
                                        listarcnul(3);
                                        if p= nil then stoprecharc:='o'
                                            else stoprecharc:='n';
                                        while stoprecharc<>'o' do
                                            begin
                                    if (p^.alpha = debut) and (p^.omega = fin)
                                    then begin listarcnul(7);
                                               stoprecharc:='o';
                                         end
                                    else if p^.avant<>nil then p:=p^.avant
                                                          else stoprecharc:='o';
                                            end;
                                        end;
                                   end
                               else if (on = 'n') or (on = 'N')
                                  then stopon:='o'
                                  else
                          writeln(tty,'veuillez repondre o pour oui et n pour non');
                               end;

                            stopdon:='n';
                            while stopdon<>'o' do
                            begin
        writeln(tty,'voulez-vous modifier l''elasticite attachee a cet arc ?');
        readln(tty);
        read(tty,don);
        if (don = 'o') or ( don = 'O')
        then begin
             stopdon:='o';
             stoptauxmod:='n';
             while stoptauxmod<>'o' do
                 begin
                 write(tty,'l''ancienne valeur de l''elasticite ');
                 writeln(tty,' etait : ',tamp.taux);
                 writeln(tty,'donnez la nouvelle valeur de l''elasticite ');
                 readln(tty);
                 read(tty,tauxmod);
                 validmesure(tauxmod);
                 if correct = 'o' then stoptauxmod:='o';
                 end;
             end
        else if (don='n') or ( don = 'N') then stopdon:='o'
            else writeln(tty,'veuillez repondre o pour oui et n pour non');
                            end;
                            if (delmod = tamp.del) and (tauxmod = tamp.taux)
                                then marque := 1

{* s''il n''y a pas eu de changement, il est inutile de recopier schema2 *}
{* dans schema1 *}

                                    else if delmod = 0 then
                                        begin


{* si le nouveau delai est nul, alors on doit introduire la fleche *}
{* dans la liste des arcs de poids nul, et incrementer nbarcnul d''une unite *}

                                            nbarcnul:= nbarcnul+1;
                                            listarcnul(1);
```

```
                                    end;
                                buf.ori := tamp.ori;
                                buf.ext := tamp.ext;
                                buf.del := delmod;
                                buf.taux := tauxmod;
                                schema2^ := buf;
                                put(schema2);
                            end
                        else begin schema2^ := tamp;
                                   put(schema2);
                             end;
                        get(schema);
                        if eof(schema) then arretlect := 'o';
                    end;
                close(schema);
                close(schema2);
                if modific = 'n'
                then begin
                     write(tty,' cette fleche n''existe pas ,');
                     write(tty,' vous pouvez utiliser la premiere operation');
                     writeln(tty,' pour l''introduire ')
                     end
                else if marque = 0 then recopiedau;
                     end;
            end;
        end;


    procedure modsom;
{* ----------------- *)

{* cette procedure permet de modifier la valeur allouee a un sommet *}

    var sommod : sommet;
        valmod : real;
        stopsommod : char;


    begin
    stopsommod:='n';
    while stopsommod<>'o' do
       begin
       writeln(tty,'quel est le sommet dont vous voulez modifier la valeur ?');
       readln(tty);
       read(tty,sommod);
       if (sommod = help1)or(sommod = help2) then help(14)
           else stopsommod:='o';
       end;

    {* on recherche si ce sommet existe bien, sinon on a termine! *}

    teste:=sommod;
    validsommet(3);
    if correct = 'o'
    then begin
         writeln(tty,'voici l''ancienne valeur de',sommod,':',l^.val);
         write(tty,'quelle est la nouvelle valeur que vous desirer');
         writeln(tty,' attacher a ',sommod,' ?');
```

```
        readln(tty);
        read(tty,valmod);
        l^.val:= valmod;
        end;

  end;



{* corps de la procedure modification *}
{* ********************************** *}


    begin
    arretmenu:= 'n';
    while arretmenu <> 'o' do
     begin
     write(tty,'faites votre choix parmi les operations suivantes,');
     writeln(tty,' tapez le numero en regard de l''operation desiree');
     writeln(tty,'1 : ajouter une fleche dans le schema');
     writeln(tty,'2 : retirer une fleche du schema');
     writeln(tty,'3 : retirer un sommet du schema');
     write(tty,'    cette operation retire egalement tout arc incident');
     writeln(tty,' au sommet que l''on retire');
     writeln(tty,'4 : modifier les valeurs d''un arc');
     writeln(tty,'5 : modifier les valeurs d''un sommet');
     readln(tty);
     read(tty,n);
     if (n<1) or (n>5)
       then writeln(tty,'veuillez donner une valeur comprise dans le menu')
       else arretmenu := 'o';
     end;
    case n of

    1 : ajfleche;
    2 : retfleche;
    3 : retsom;
    4 : modfleche;
    5 : modsom;
    end;

    end;


  procedure demaffich;
{* ------------------- *}

  var stopaffich,affich : char;

  begin
  stopaffich:='n';
  while stopaffich<>'o' do
      begin
      writeln(tty,'voulez vous voir les resultats de vos modifications ?');
      readln(tty);
      read(tty,affich);
      if (affich = 'h') or (affich = 'H')
          then help(15)
```

```
            else
            if (affich = 'o') or (affich = 'O')
                then begin
                     stopaffich := 'o';
                     affichage;
                     end
                else
                if (affich = 'n')or(affich = 'N')
                    then stopaffich := 'o'
                    else
                    writeln(tty,'veuillez repondre o pour oui, et n pour non ');

        end;
    end;


{* corps de la procedure demandemodification *}
{* ********************************** *}


    begin
    stopdemande:='n';
    while stopdemande <>'o'do
        begin
        writeln(tty,'voulez vous modifier le schema donne ?');
        readln(tty);
        read(tty,answ);
        if (answ = 'h')or(answ = 'H')
            then help(10)
            else
            if (answ = 'n') or (answ = 'N')
                then stopdemande:='o'
                else
                if (answ ='o') or (answ = 'O')
                    then  begin
                          modification;
                          demaffich;
                          end
                    else
                     writeln(tty,'voudriez vous repondre o pour oui,et n pour non');

        end;
    end;



    procedure demandeexecution;
{* ------------------------- *}

{* cette procedure appelle une procedure d''execution de la procedure *}
{* memoire pour autant que l''utilisateur le souhaite *}


      procedure execution;
     {* ------------------- *}
```

```
{* cette procedure appelle la procedure memoire pour autant que *)
{* les conditions necessaires a son application soient reunies : *}
{* le fichier schema ne peut etre vide, *}
{* et il n''existe pas de circuit de poids total nul *}

   var rep,arretrep : char;


    procedure memoire;
    {* ------------------ *}

    {* cette procedure etablit l''algorithme presente dans le memoire *}

       type ptsom = ^lignecol;
            lignecol = record
                           nom : sommet;
                           elast : real;
                           noligne,nocol : integer;
                           avligne,avcol : ptsom;
                       end;


        var s,pcdtcol,pcdtligne,somcvd : ptsom;
            I,ii,j,TI : integer;
            C : array [0..lindeli] of integer;
            arretp,arretd,arrett,arretq,arretrech,arretti : char;
            cvd,X : sommet;
            imp,tho,initval : real;
            stopligne,stoplect,finlect,arretcvd : char;
            stopdem,dem : char;
            passom,existe :integer;


    procedure gestmat(n : integer);
    {* ------------------------------ *}

    {* cette procedure gere une liste d''elements groupes *}
    {* sous forme de matrice *}


       begin
       case n of

       0 : {* initialisation des pointeurs *}
           begin somcvd:= nil;
                 pcdtcol := nil;
                 pcdtligne := nil;
           end;

       1 : {* ecriture du cvd *}
           begin new(s);
                 somcvd := s;
                 s^.nom := cvd;
                 s^.elast := 1;
                 s^.noligne := 0;
                 s^.nocol := 1;
                 s^.avligne := nil;
                 s^.avcol := nil;
```

```
                pcdtligne := s;
                pcdtcol := s;
        end;

    2 : {* ecriture du premier element d'une nouvelle ligne *}
        begin s := somcvd;
        while s^.avcol<>nil do s:= s^.avcol;
        new(s);
        s^.nom := tamp.ext;
        s^.elast := tho * tamp.taux;
        s^.noligne := I;
        s^.nocol := 1;
        s^.avligne := nil;
        s^.avcol := nil;
        pcdtcol^.avcol := s;
        pcdtcol := s;
        pcdtligne := s;
        end;

    3 : {* ecriture d'un element a la fin d'une ligne *}
        begin s:= somcvd;
        while s^.avcol<> nil do s:= s^.avcol;
        while s^.avligne<>nil do s:= s^.avligne;
        new(s);
        s^.nom := tamp.ext;
        s^.elast := tho * tamp.taux;
        s^.noligne := I;
        s^.nocol := C[I];
        s^.avligne := nil;
        s^.avcol := nil;
        pcdtligne^.avligne := s;
        pcdtligne := s;
        end;

    4 : {* avancer sur la premiere colonne *}
        if s^.avcol <> nil then s:= s^.avcol;


    5 : {* avancer sur une ligne *}
        if s^.avligne<> nil then s:= s^.avligne;


    6 : {* se positionner au depart *}
        s := somcvd;
end;
end;


procedure positionner(l,c : integer);
{* ------------------------------------ *}

{* cette procedure permet de se positionner en l''element (l,c) *}
{* de la matrice *}

begin
gestmat(6);
while s^.noligne<>l do gestmat(4);
while s^.nocol<> c do gestmat(5);
end;
```

```
   procedure Pboucle;
{* ------------------ *}

{* cette procedure effectue la premiere boucle de l''algorithme *}
{* presente dans le memoire,*}
{* c''est a dire la boucle sur les valeurs de I,le delai *}
{* allant de 0 a TI *}


   procedure Dboucle;
{* ------------------ *}

{* cette procedure effectue la deuxieme boucle de l''algorithme *}
{* presente dans le memoire,*}
{* c''est a dire la boucle sur les valeurs de i,*}
{* et parcourant les lignes precedentes a la ligne I *}


   procedure Tboucle;
{* ------------------ *}

{* cette procedure effectue la troisieme boucle de l'algorithme *}
{* presente dans le memoire,*}
{* c'' est a dire la boucle sur les valeurs de j,*}
{* et parcourt ainsi les elements de la ligne I-ii *}


   procedure Qboucle;
{* ------------------ *}

{* cette procedure effectue la quatrieme boucle de l''algorithme *}
{* presente dans le memoire,*}
{* c''est a dire qu''on parcourt les suivants du sommet X *}
{* pour selectionner ceux qui le sont de par un arc de delai i *}

   var pasx : integer;

   begin
   reset(schema);
   if eof(schema) then arretq:='o'
       else arretq := 'n';
   pasx := 0;
   while arretq <> 'o' do
       begin
       tamp := schema^;
       if tamp.ori=X
           then begin if tamp.del = ii
                then begin

{* s''il n''y a pas encore d''element dans la Ieme ligne de la matrice *}
{* alors on appelle gestmat2 car cela permet de mettre a jour le pointeur *}
{* colonne, sinon on appelle gestmat3 *}
```

```
                                c[I] := c[I] +1;
                                if c[I] = 1 then gestmat(2)
                                    else gestmat(3);

{* le nombre d''elements sur la Ieme ligne s''incremente d''une unite *}

                                end;

                            pasx :=1;
                            get(schema);
                            if eof(schema) then arretq := 'o';
                            end

                        else if pasx <> 0 then arretq := 'o'
                            else begin get(schema);
                                if eof(schema) then arretq :='o';

{* on s''arrete soit lorsqu''on arrive au bout du fichier schema, *}
{*  soit quand on a fini d''explorer les suivants du sommet X *}

                                end;
                    end;

                close(schema);
                end;


            {* corps de la troisieme boucle *}
            {* **************************** *}

                begin
                    arrett := 'n';
                    while arrett <> 'o' do
                        begin
                        positionner(I-ii,j);
                        X:= s^.nom;
                        tho := s^.elast;
                        gboucle;
                        j:=j+1;
                        if j > c[I-ii] then arrett := 'o';

{* on s''arrete lorqu''il n''y a plus d''elements sur la ligne I-ii *}

                        end;
                end;


            {* corps de la deuxieme boucle *}
            {* **************************** *}

                begin
                arretd:='n';
                while arretd <> 'o' do
                    begin
```

```
                if c[I-ii] > 0 then begin j:=1;
                                          Dboucle;
                                    end;
                ii:=ii-1;
                if ii < 0 then arretj:='o';
                end;

{* on s''arrete lorsqu''on a parcouru toutes les lignes inferieures *}
{* ou egales a la ligne I *}

            end;


        {* corps de la premiere boucle *}
        {* *************************** *}

            begin
            arretp := 'n';
            while arretp <> 'o' do
                begin
                ii := I;
                if I = 0 then c[I] := 1 else c[I] := 0;
                Dboucle;
                I := I+1;
                if I > TI then arretp := 'o';

                {* on s''arrete lorsqu''on a depasse le temps imparti *}

                end;
            end;


            procedure resultaffich;
        {* ---------------------- *}

        {* Cette procedure affiche les resultats de la simulation, *}
        {* ligne par ligne, sans recapituler les influences sur un sommet *}

            var stopligne,stoplect : char;

            begin
            writeln(tty,'  DETAIL des calculs de mesure des influences ');
            writeln(tty,'  -------');
            gestmat(6);
            if s = nil then stoplect := 'o'
                else stoplect := 'n';
            while stoplect<>'o' do
                begin
        writeln(tty,'voici les elements influences apres un delai ',s^.noligne);
        stopligne:='n';
        while stopligne<>'o' do
            begin
            writeln(tty,s^.nom);
            writeln(tty,'avec un coefficient : ',s^.elast);
            writeln(tty,'donc une influence de :',imp*s^.elast);

            rechvalsom(s^.nom);
```

```
        writeln(tty,'de valeur initiale : ',l^.val);
        writeln(tty,'de nouvelle valeur : ',l^.val+(imp*s^.elast));

        if s^.avligne = nil then stopligne :='o'
          else s:= s^.avligne;
        end;
      positionner(s^.noligne,1);
      if s^.avcol = nil then stoplect := 'o'
        else s:= s^.avcol;
          end;
        end;


        procedure recapaffich;
      {* ---------------------- *}

      {* Cette procedure affiche les resultats de la simulation *}
      {* en recapitulant pour chaque sommet les influences qu''il a subies*}
      {* dans un meme delai *}

        var stoplect,stopligne,stopliste : char;

        begin
        writeln(tty);
        write(tty,'  TABLEAU RECAPITULATIF par delai et par sommet');
        writeln(tty,' des valeurs des influences ');
        writeln(tty,'  ----------------------');
        gestmat(5);
        if s = nil then stoplect:='o' else stoplect :='n';
        while stoplect <> 'o' do
            begin
            stopligne := 'n';
            while stopligne <> 'o' do
                begin
                rechvalsom(s^.nom);
                l^.valaj:=l^.valaj+(imp*s^.elast);
                if s^.avligne = nil then stopligne := 'o'
                  else s:= s^.avligne;
                end;
            writeln(tty);
            write(tty,' voici les sommets influences apres ');
            writeln(tty,' un delai : ',s^.noligne);
            writeln(tty);
       writeln(tty,'   sommet    !  val. init !  var.totale !  nouv. valeur ');
       writeln(tty,'-------------!------------!-------------!----------------');
            listsomval(3);
            if l = nil then stopliste := 'o' else stopliste := 'n';
            while stopliste <> 'o' do
                begin
                if l^.valaj <> 0
                  then begin
writeln(tty,l^.som:12,'!',l^.val:12,'!',l^.valaj:12,'!',(l^.valaj+l^.val):12);
                       l^.valaj:=0;
                       end;
                   if l^.svt = nil then stopliste :='o'
                     else l:= l^.svt;
                end;
            positionner(s^.noligne,1);
            if s^.avcol = nil then stoplect :='o'
                  else s:= s^.avcol;
```

```
                end;
        end;

        procedure printdemande;
    {* ---------------------- *}

        var rep,stoprep : char;

        procedure printresultat;
    {* ----------------------- *}

        var stoplect,stopligne,stopliste,stopdem,den : char;

        procedure schemaffich;
    {* ---------------------- *}

        var stop1,stoplist : char;

        begin
        reset(schema);
        if eof(schema) then stop1:='o' else stop1:= 'n';
        writeln(resultat);
        write(resultat,'Voici la liste des fleches constituant ');
        writeln(resultat,' le schema au moment de la simulation');
        write(resultat,'-----------------------------------');
        writeln(resultat,'-----------------------------------');
        writeln(resultat);
  writeln(resultat,'  origine   ! extremite  !   delai    ! coefficient ');
  writeln(resultat,'------------!------------!------------!-------------');
        if stop1 = 'o'
        then writeln(resultat,'ce schema ne contient aucune fleche');
        while stop1<>'o' do
        begin
        temp:=schema^;
writeln(resultat,temp.ori:12,'!',temp.ext:12,'!',temp.del:12,'!',temp.taux:12);
writeln(resultat,'------------!------------!------------!-------------');
        get(schema);
        if eof(schema) then stop1:='o';
        end;
        close(schema);

        listsomval(3);
        if l = nil then stoplist:='o'
        else begin
             stoplist:='n';
             writeln(resultat);
             write(resultat,'voici la liste des centres de variation ');
             write(resultat,' et leur valeur initiale associee, au ');
             writeln(resultat,' moment de la simulation');
             while stoplist<>'o' do
                 begin
                 writeln(resultat,l^.son,' ; ',l^.val);
                 if l^.dvt <> nil then l:=l^.dvt
                     else stoplist := 'o';
                 end;
             end;
        end;
```

```
        procedure printdetail;
    {* ---------------------- *}

        var stopligne,stoplect : char;

        begin
        gestmat(6);
        if s = nil then stoplect := 'o' else stoplect:='n';
        while stoplect<>'o' do
            begin
            write(resultat,'voici les elements influences apres ');
            writeln(resultat,' un delai ',s^.noligne);
            stopligne :='n';
            while stopligne <>'o' do
                begin
                writeln(resultat);
                writeln(resultat,s^.nom);
                writeln(resultat,'avec un coefficient : ',s^.elast);
               writeln(resultat,'donc une influence de : ',imp*s^.elast);
                rechvalsom(s^.nom);
                writeln(resultat,'de valeur initiale : ',l^.val);
     writeln(resultat,'de nouvelle valeur : ',l^.val+(imp*s^.elast));
                if s^.avligne = nil then stopligne :='o'
                    else s:=s^.avligne;
                end;
            positionner(s^.noligne,1);
            if s^.avcol = nil then stoplect :='o'
                else s:=s^.avcol;
            end;
        writeln(resultat);
        write(resultat,'                                    fin de ');
        writeln(resultat,' la ',nosim,'eme simulation');
        writeln(resultat);
        writeln(resultat);
writeln(resultat,'                    ---------------------------');
        writeln(resultat);
        writeln(resultat);
        end;


    {* corps de la procedure printresultat *}
    {* *********************************** *}


        begin
        append(resultat);
        nosim:=nosim+1;
        write(resultat,'Voici les resultats de votre ');
        writeln(resultat,nosim,'eme simulation.');
        writeln(resultat);
        writeln(resultat);
        schemaffich;
        writeln(resultat);
    writeln(resultat,'Voici les conditions initiales de votre simulation ');
        writeln(resultat);
        writeln(resultat,'cvd : ',cvd);
        writeln(resultat);
        writeln(resultat,'temps imparti : ',TI);
```

```
                writeln(resultat);
                writeln(resultat,'impulsion de depart : ',imp);
                writeln(resultat);
              writeln(resultat,'Voici le tableau recapitulatif des influences ');
                gestmat(6);
                if s = nil then stoplect := 'o' else stoplect := 'n';
                while stoplect <> 'o' do
                    begin
                    stopligne := 'n';
                    while stopligne <> 'o' do
                        begin
                        rechvalsom(s^.nom);
                        l^.valaj := l^.valaj+(imp*s^.elast);
                        if s^.avligne = nil then stopligne :='o'
                          else s:=s^.avligne;
                        end;
                    writeln(resultat);
 writeln(resultat,'voici les sommets influences apres un delai : ',s^.noligne);
 writeln(resultat);
 writeln(resultat,'    sommet    !  val. init !  var.totale !  nouv.valeur ');
 writeln(resultat,'--------------!-------------!-------------!---------------');
                    listsomval(3);
                    if l = nil then stopliste :='o' else stopliste :='n';
                    while stopliste <>'o' do
                        begin
                        if l^.valaj<>0
                        then begin
writeln(resultat,l^.som:12,'!',l^.val:12,'!',l^.valaj:12,'!',(l^.valaj+l^.val):12);
writeln(resultat,'--------------!-------------!-------------!---------------');
                             l^.valaj := 0;
                             end;
                        if l^.dvt = nil then stopliste :='o'
                          else l:= l^.dvt;
                        end;
                    positionner(s^.noligne,1);
                    if s^.avcol = nil then stoplect :='o'
                        else s:=s^.avcol;
                    end;
                writeln(resultat);
                write(resultat,'                                        ----------------------');
                write(resultat,'----------------------------------------');
                writeln(resultat,'--------------');
                stopdem:='n';
                while stopdem<>'o' do
                    begin
        writeln(tty,'Desirez vous egalement l''impression des details ?');
        readln(tty);
        read(tty,dem);
        if (dem = 'o') or (dem = 'O') then
                begin
                writeln(resultat);

                write(resultat,'voici les operations qui permirent d''obtenir ');
                writeln(resultat,'le tableau ci dessus');
                writeln(resultat);
                printdetail;
                stopdem:='o';
                end
        else if (dem = 'N') or (dem = 'n')
            then stopdem:='o'
```

```
        else begin write(tty,'veuillez repondre o pour oui');
                   writeln(tty,' et n pour non ');
             end;
          end;
       close(resultat);
       writeln(tty,'ecriture dans fichier resultat ok!');
       end;


    {* corps de la procedure printdemande *}
    {* ********************************** *}


    begin
    stoprep := 'n';
    while stoprep<>'o' do
       begin
writeln(tty,'voulez vous imprimer les resultats de cette simulation ?');
       readln(tty);
       read(tty,rep);
       if (rep ='h')or(rep ='H') then help(16)
       else
            if (rep ='o') or (rep ='O')
            then begin
                 printresultat;
                 stoprep:='o';
                 end
            else if (rep = 'n')or(rep ='N') then stoprep := 'o'
     else writeln(tty,'veuillez repondre o pour oui et n pour non ');
                 end;
    end;


    {* corps de la procedure memoire *}
    {* ***************************** *}


    begin
    arretcvd:='n';
    while arretcvd<>'o' do
       begin
       writeln(tty,'quel est le centre de variation de depart ?');
       readln(tty);
       read(tty,cvd);
       if (cvd = help1)or(cvd = help2) then help(3)
            else begin

                      {* on verifie que ce sommet cvd existe bien *}

                 teste:=cvd;
                 validsommet(3);
                 if correct = 'o' then arretcvd:='o';
                 end;
       end;


    destmat(0);
    destmat(1);

    arretti:='n';
```

```
    while arretti<>'o' do
        begin
        writeln(tty,'quel est le temps imparti ?');
        readln(tty);
        read(tty,TI);

        {* le temps imparti doit etre compris entre 0 et limdel *}

        validtemps(TI);
        if correct = 'o' then arretti:='o';
        end;

    writeln(tty,'quelle est l''impulsion de depart sur le cvd ?');
    readln(tty);
    read(tty,imp);

    {* l''impulsion donnee au depart est quelconque *}

    I := 0;
    Pboucle;
    recapaffich;
    stopdem:='n';
    while stopdem<>'o' do
        begin
        writeln(tty);
        write(tty,'desirez vous voir le detail des demarches ');
        writeln(tty,'qui ont permis d''etablir ce tableau ?');
        readln(tty);
        read(tty,dem);
        if (dem = 'h') or (dem = 'H')
            then help(17)
            else begin
                 if (dem = 'o') or (dem ='O')
                     then begin stopdem:='o';
                                resultaffich;
                          end
                     else if (dem='n')or(dem='N')
                          then stopdem:='o'
        else writeln(tty,'veuillez repondre o pour oui, et n pour non');
                 end;
        end;
    printdemande;

    end;


    procedure rechcircnul;
{*  ---------------------  *}

{* cette procedure recherche les circuits nuls possibles *}

    type pointe =^lelement;
         lelement = record
                        elmt:sommet;
                        num : integer;
                        lavant,larriere : pointe
                    end;

         pteImt = ^elemcirc;
```

```
        elemcirc = record
                        mot : sommet;
                        nbe : integer;
                        av,ar : ptelmt;
                   end;

        ptmarque = ^marquage;
        marquage = record
                        mark,number : integer;
                        forw,back : ptmarque;
                   end;


   var no,u,k,w,v : integer;
       e,ledepart,larrivee,eavancer,ereculer : pointe;
       c,dep,arriv,cavancer,creculer : ptelmt;
       m,departure,arrival,mavancer,mreculer : ptmarque;
       lepremier,first,prem : char;
       stopcirc : char;


   procedure listecircnul(n:integer);
{* ------------------------------- *}

{* cette procedure gere une liste des sommets *}
{* pouvant constituer un circuit de poids total nul *}

   begin case n of

   0 : {* initialisations des pointeurs *}
       begin ledepart:=nil;
             larrivee:=nil;
             eavancer :=nil;ereculer := nil;
             lepremier := 'o'
       end;

   1 : {* insertion d''un element en fin de liste *}
       begin if lepremier = 'o'
        then begin e := larrivee;
                   new(e);
                   ledepart:= e;
                   e^.elmt:=p^.alpha;
                   e^.num := u;
                   e^.lavant := nil;
                   e^.larriere := larrivee;
                   lepremier := 'n';
                   larrivee:=e;
             end
        else begin e:= larrivee;
                   new(e);
                   larrivee^.lavant:=e;
                   e^.elmt:=p^.onega;
                   e^.num := u;
                   e^.lavant := nil;
                   e^.larriere := larrivee;
                   larrivee:=e;
             end;
       end;
```

```
    2 : {* effacement d''un element en fin de liste *}
        begin e:=larrivee;
              larrivee := larrivee^.darriere;
              dispose(e);
              if larrivee <> nil then larrivee^.lavant := nil ;
              e:=larrivee;
        end;

    3 :{* positionnement sur le prenier element de la liste *}
       begin eavancer:= ledepart;e:=eavancer;end;

    4 :{* positionnement sur le dernier element de la liste *}
       begin ereculer:= larrivee;e:=ereculer;end;

    5 :{* lecture de l''element suivant *}
       begin eavancer := eavancer^.lavant;e:=eavancer;end;

    6 :{* lecture de l''element precedent *}
       begin ereculer := ereculer^.larriere;e:=ereculer;end;
          end;
    end;



    procedure gestelnt(n:integer);
{*  ------------------------------*}

{* cette procedure gere une liste des elements constituant *}
{* des circuits de poids total nul *}

    begin case n of

    0 : {* initialisation des pointeurs *}
        begin dep := nil;arriv := nil; cavancer := nil;
              creculer := nil; prem := 'o';
        end;

    1 : {* insertion d''un element en fin de liste *}
        begin c := arriv;
              new(c);
              if prem = 'o' then begin dep := c;
                                       prem :='n';
                                 end
              else arriv^.av := c;
              c^.not := e^.elnt;
              c^.nbe := k;
              c^.av := nil;
              c^.ar := arriv;
              arriv := c;
        end;

    2: {* effacement d''un element en fin de liste *}
       begin c := arriv;
             arriv := arriv^.ar;
             dispose (c);
             if arriv <> nil then arriv^.av := nil ;
             c := arriv;
       end;
```

```
            3:{* positionnement sur le premier element de la liste *}
              begin cavancer := dep; c := cavancer; end;

            4:{* positionnement sur le dernier element de la liste *}
              begin creculer := arriv; c := creculer; end;

            5: {* lecture de l''element suivant *}
               begin cavancer := cavancer^.av;c := cavancer;end;

            6: {* lecture de l''element precedent *}
               begin creculer:= creculer^.ar;c:= creculer; end;

                end;
            end;


            procedure gestmarq(n:integer);
        {* ---------------------------- *}


{* cette procedure gere une liste des longueurs des circuits nul *}
{* places dans gestelmt , ce qui permet de ne comparer entre eux *}
{* que des circuits de meme longueur *}

            begin case n of

            0 :{* initialisation des pointeurs *}
               begin departure := nil; arrival := nil;
                     mavancer := nil; mreculer := nil;
                     first := 'o';
               end;

            1 :{* insertion d''un element en fin de liste *}
               begin m := arrival;
                     new(m);
                     if first = 'o' then begin first := 'n';
                                               departure := m;
                                         end
                        else arrival^.forw := m;
                     m^.mark := v;
                     m^.number := w;
                     m^.forw := nil;
                     m^.back := arrival;
                     arrival := m;
               end;

            2 : {* effacement d''un element en fin de liste *}
                begin m := arrival;
                      arrival := arrival^.back;
                      dispose(m);
                      if arrival <> nil then arrival^.forw := nil ;
                      m:= arrival;
                end;

            3 : {* positionnement sur le premier element de la liste *}
                begin mavancer := departure;
                      m := mavancer;
                end;
```

```
            4 : {* positionnement sur le dernier element de la liste *}
                begin mreculer := arrival;
                      m := mreculer;
                end;

            5 : {* lecture de l''element suivant *}
                begin mavancer := mavancer^.forw;
                      m := mavancer;
                end;

            6 : {* lecture de l''element precedent *}
                begin mreculer := mreculer^.back;
                      m := mreculer;
                end;

              end;
        end;


    procedure explore;
{* ---------------- *}

{* partant d''un arc donne, cette procedure explore les chemins en decoulant *}
{* pour y chercher un eventuel circuit nul *}

        var elem : sommet;
            i,j,indic,circ : integer;

    procedure verifcirc;
{* ------------------- *}

{* cette procedure verifie s''il n''existe pas de circuit *}
{* parmi les elements de listcircnul *}

        var ad1,ad2,genoeg,remember : integer;
            comp : char;

    procedure comparaison;
{* --------------------- *}

{* cette procedure compare les circuits obtenus *}
{* afin d''elimmer les redondances *}

        var kk,nn : sommet;
            x,y,note,lg,ok,egal : integer;


        begin
        x :=0;
        y :=0;
        note := m^.number;
        lg := m^.mark;
        ok :=0;
```

```
            gestmarq(3);
            while m^.number < note do
                begin x := x + m^.mark;
                      gestmarq(5);
                end;
            x := x+1;
            adi := x;

{* adi est l''adresse(numero) du premier element du premier circuit *}
{* que l''on compare *}

            gestelmt(4);
            y := c^.nbe;
            gestmarq(4);
            y := y - m^.mark;
            ad2 := y+1;y:=ad2;

{* ad2 est l''adresse(numero) du premier element du deuxieme circuit compare *}

            egal :=0;
            gestelmt(3);
            while c^.nbe <> x do gestelmt(5);
            kk := c^.mot;

            gestelmt(3);
            while c^.nbe <> y do gestelmt(5);

{* on recherche si, dans le deuxieme circuit ,*}
{* il existe un element identique au premier element du premier circuit *}

            while (y < ad2+lg) and (egal = 0) do
                begin if c^.mot = kk
                    then egal := 1
                        else begin
                             y := y+1;
                             gestelmt(5);
                             end;
                end;

{* si un tel element existe, on effectue la comparaison sur les autres *}
{* elements des deux circuits. et cela tant qu''il y a egalite et tant *}
{* qu''on arrive pas a la fin du circuit *}

            if egal <> 0
            then begin hh := c^.mot;
                while (hh = kk) and (x < adi+lg) do
                    begin x := x+1; ok := ok+1;
                          if y+1 < ad2+lg then y := y+1
                              else y := ad2;
                          gestelmt(3);
                          while c^.nbe <> x do gestelmt(5);
                          kk := c^.mot;
```

```
                        gestelmt(3);
                        while c^.nbe <> y do gestelmt(5);
                        hh := c^.mot;
                    end;


{* ok s''incrementait chaque fois qu''il y avait egalite entre les elements *}
{* de chacun des ciruits, des lors, si ok vaut  lg,la longueur des circuits,*}
{* on en deduit que ceux-ci sont egaux *}

                if ok = lg then comp := '+'
                    else comp := '-';
                end
            else comp:= '-';
            end;



            procedure effacement;
        {*  ------------------- *}

            {* cette procedure efface les circuits deja nommes *}
            {* si la comparaison s''est avereee positive, alors on efface *}
            {* de la liste gestelemt, les elements du deuxieme circuit *}
            {* redondant par rapport au premier circuit de la comparaison *}

            var indic,long : integer;


            begin
            indic := 0;
            gestmarq(4);
            long := m^.mark;
            k := k-long;
            while indic < long do
                begin
                gestelmt(4);
                gestelmt(2);
                indic := indic +1;
                end;
            gestmarq(4);
            gestmarq(2);
            w := w-1;


{* w indique le nombre de circuits existants,*}
{* on le decremente a chaque fois qu''on efface un circuit *}

            end;


            procedure ecritcircnul;
        {*  ---------------------- *}

{* cette procedure ecrit les circuits de poids total nul a l''ecran *}

            var utile : sommet;

{* utile sert a indiquer le sommet origine, et donc extremite du circuit *}
```

```
            begin
            indiccirc:=1;
            gestelmt(3);
            while c^.nbe <> ad2 do gestelmt(5);
            utile := c^.mot;
            write (tty,'il existe un circuit de poids total nul :');
            while c^.av <> nil do
                begin write (tty,c^.mot,',');
                      gestelmt(5);
                end;
            write(tty,c^.mot,',');
            write (tty,utile);
            writeln(tty);
            end;


        {* corps de la procedure verifcirc *}
        {* ******************************** *}

            begin
            circ := 0;
            while j < u do
                begin
                listecircnul(3);
                while e^.num <> j do listecircnul(5);

{* chaque nouvel element rajoute dans listcircnul, est comparé a tous *}
{* les elements qui s''y trouvent deja, si la comparaison est positive,*}
{* on en deduit qu''il existe un circuit *}

                if E^.elmt = P^.omega
                    then begin
                         circ := 1;
                         gestelmt(4);
                         if c<>nil then begin ad2 := c^.nbe ;
                                              ad2 := ad2+1;
                                        end
                                   else ad2 := 1;
                         j:=u;
                         v :=0;

{* on place le circuit trouve dans gestelmt *}

                         while e^.lavant <> nil do
                             begin k:=k+1;v := v+1;
                                   gestelmt(i);
                                   listecircnul(5);
                             end;
                         w := w+1;

{* w indiquant le nombre de circuits nuls trouves,s''incremente d''une unite *}

                         gestmarq(1);
                         gestmarq(4);
```

```
{* on place la longueur de ce nouveau circuit dans gestmarq *}

                        comp := '-';genoeg := arrival^.mark;
                        while (m^.back<> nil) and (comp ='-') do
                            begin
                            gestmarq(6);remember := m^.number;

                            if m^.mark = genoeg

                    {* si la longueur du circuit designe par sa marque *}
                    {* est egale a la longueur du circuit que l''on vient *}
                    {* de decouvrir, alors on effectue la comparaison *}

                        then begin
                             comparaison;
                             if comp = '+'
                               then effacement;
                             end;
                        gestmarq(4);
                        while m^.number <> remember do gestmarq(6);
                    end;
                    if comp = '-' then ecritcircnul;

{* si la comparaison est negative, c''est a dire si ce circuit n''est *}
{* pas equivalent a un autre deja trouve, on l''affiche a l''ecran, *}
{* sans l''effacer de gestelmt car il doit pouvoir servir de point *}
{* eventuel de comparaison *}

                                        end
                                        else j := j+1;
                    end;
                end;


            {* corps de la procedure explore *}
            {* ***************************** *}

            {* cette procedure, reccursive, cherche a etablir des circuits *}
            {* de poids total nul *}


               begin
               elem := P^.omega;
               i := 1;
               while i<= nbarcnul do
                   begin
                   listarcnul(3);
                   while p^.numero <> i do listarcnul(5);

                   if p^.alpha = elem
                       then begin j:=1;
                                  u:=u+1;
                                  listecircnul(1);
                                  verifcirc;
                                  if circ = 0
                                      then begin explore;
                                                 u := u-1;
                                                 listecircnul(4);
                                                 listecircnul(2);
```

```
                                        listecircnul(4);
                                        elem := larrivee^.elmt;
                                    end
                                else

{* si on a trouve un circuit, alors, on revient en arriere dans listcircnul *}

                                    begin u := u-1;
                                        indic:=1;
                                        listecircnul(4);
                                        listecircnul(2);
                                        listecircnul(4);
                                        elem := larrivee^.elmt;
                                    end;
                            end;
                        i:=i+1;
                    end;
                end;


            {* corps de la procedure rechcircnul *}
            {* ********************************* *}

                begin
                no:=1;
                k:=0;w:=0;
                gestelmt(0);
                gestmarq(0);
                while no <= nbarcnul do
                    begin
                    listecircnul(0);
                    listarcnul(3);
                    stopcirc:='n';
                    while stopcirc<>'o' do
                        begin
                        if p^.numero <>no
                            then if p^.avant<>nil
                                then listarcnul(5)
                                    else stopcirc:='o'
                                else stopcirc:='o';
                        end;
                    if p^.numero = no
                        then begin
                            u:=1;
                            listecircnul(1);
                            u:=2;
                            listecircnul(1);
                            explore;

{* on appelle la procedure explore pour chaque element de listarcnul *}

                            no := no+1;
                            end;

                    end;
                end;
```

```
{* corps de la procedure execution *}
{* ------------------------------- *}

   begin
   if nart = 0 then writeln(tty,'schema vide ?')
   else begin
        indiccirc:=0;
        if nbarcnul>1 then rechcircnul;
        if indiccirc = 0
            then begin
                 stopmemoire :='n';
                 while stopmemoire <>'o' do
                     begin
                     memoire;
                     arretrep := 'n';
                     while arretrep<>'o' do
                         begin
     writeln(tty,'voulez-vous reexecuter le programme avec le meme schema ?');

{* l''utilisateur peut vouloir recommencer la simulation sur le meme schema *}
{* mais pour une valeur du cvd differente, ou pour un autre temps imparti *}
{* ou encore pour une impulsion differente *}

     readln(tty);
     read(tty,rep);
     if (rep = 'h')or(rep ='H') then help(11)
         else begin
              if (rep = 'n') or (rep = 'N') then begin
                                                 arretrep:='o';
                                                 stopmemoire:='o';
                                                 end
            else if (rep = 'o') or (rep = 'O')
               then arretrep:='o'
             else writeln(tty,'veuillez repondre o pour oui, et n pour non');
            end;
                                 end;
                             end;
                    end;
             end;
             end;


{* corps de la procedure demandeexecution *}
{* -------------------------------------- *}

   begin
   stopreex:= 'n';
   while stopreex<>'o' do
       begin
       write(tty,'voulez-vous executer le programme ');
       writeln(tty,' avec le schema dernierement entre ?');
       readln(tty);
       read(tty,reex);
       if (reex= 'h')or(reex = 'H') then help(12)
          else begin
       if (reex='n') or (reex = 'N') then begin stopreex:='o';
                                          finfin:='o';
```

```
{* si l''utilisateur ne veut plus executer le programme,*}
{* alors celui ci se termine definitivement *}

                                end
                            else if (reex ='o') or (reex = 'O')
                                then begin
                                     stopreex:='o';
                                     execution;
                                     end
              else writeln(tty,'veuillez repondre o pour oui et n pour non');
                        end;
                end;
            end;


        procedure reformation;
{* ------------------------ *}

{* cette procedure reconstruit les conditions d'execution du *}
{* programme. Elle reconstruit la liste des arcs nuls, recalcule *}
{* nbarcnul et nart, et finalement demande les valeurs des sommets *}
{* se trouvant dans le sc hema mais qui ne seraient pas dans la *}
{* liste des sommets *}

 var stopschema:char;

 begin
 stopschema:='n';
 while stopschema<>'o' do
      begin
      tamp:=schema^;
      teste := tamp.ori;
      message:='n';
      validsommet(3);
      if correct = 'n' then begin
                            vv:=tamp.ori;
                            listsomval(1);
                            end
      else begin
           teste := tamp.ext;
           validsommet(3);
           message:='n';
           if correct = 'n'
               then begin
                    vv:=tamp.ext;
                    listsomval(1);
                    end
               else begin
                    message:='o';
                    if tamp.del = 0
                    then begin
                         nbarcnul := nbarcnul +1;
                         debut := tamp.ori;
                         fin := tamp.ext;
                         listarcnul(1);
                         end;
                    nart:=nart+1;
                    get(schema);
                    if eof(schema) then stopschema:= 'o';
```

```
                    end;
            end;
        end;
end;


    procedure sauvetage;
{* ------------------- *}

{* cette procedure sauve la liste des sommets et de leurs valeurs *}
{* respectives dans fsommet *}

    var stoplist : char;

    begin
    listsomval(3);
    if l = nil then stoplist:='o' else  stoplist :='n';
    rewrite(fsommet);
    while stoplist <> 'o' do
        begin
        tampon.cv:=l^.som;
        tampon.valeur:=l^.val;
        fsommet^:=tampon;
        put(fsommet);
        if l^.dvt=nil then stoplist:='o'
            else l:=l^.dvt;
        end;
    close(fsommet);
    end;


    procedure rehabilitation;
{* ------------------------ *}

{* cette procedure reconstruit la liste des sommets *}
{* et de leurs valeurs initiales a partir du fichier fsommet *}

    var arretsomval : char;

    begin
    reset(fsommet);
    if eof(fsommet)then arretsomval:='o' else arretsomval:='n';
    valeurexiste := 1;
    while arretsomval<>'o' do
        begin
        tampon:=fsommet^;
        vv:= tampon.cv;
        valinit:= tampon.valeur;
        listsomval(1);
        get(fsommet);
        if eof(fsommet) then arretsomval:='o';
        end;
    close(fsommet);
    end;


{* corps du programme *}
{* ****************** *}
```

```
begin
writeln(tty);
writeln(tty,' Ce programme a ete realise par b.Tasiaux, ');
writeln(tty,' dans le cadre d''un memoire en informatique,');
writeln(tty,' sous la direction de J-C. Jacquemin et J. Berleur,');
writeln(tty,' annee academique 1983-1984 .');
writeln(tty);
part := 0;nbarcnul := 0;
listarcnul(0);
listsomval(0);
rewrite(resultat);
writeln(resultat);
write(resultat,'Voici les resultats des differentes simulations ');
write(resultat,' effectuees');
writeln(resultat,' lors de l''execution du programme.');
writeln(resultat);
nosim:=0;
close(resultat);

{* on ecrit l'en-tete du fichier des resultats *}

impossible := 'n';
valeurexiste := 1;
message :='o';
reset(schema);
if eof(schema)
    then begin
         close(schema);
         rewrite(schema);
         arret := 'n';
         valeurexiste := 0;
         while arret = 'n' do introduction(arret);
         end
    else begin
         rehabilitation;
         valeurexiste:=0;
         reformation;
         end;
 close(schema);
 valeurexiste := 0;

 finfin := 'n';
 while finfin <> 'o' do
     begin
         affichage;
         demandemodification;
         demandeexecution;
     end;
 sauvetage;

 write(tty,'En esperant que vous avez eu autant de plaisir');
 write(tty,' a travailler avec moi que j''en ai eprouve a ');
 write(tty,'travailler avec vous, ');
 writeln(tty,'je vous dis au revoir, et a bientot.');
 writeln(tty);
 write(tty,'Vous trouverez les resultats de vos simulations dans ');
 write(tty,'le fichier que vous avez donne en entree sous la');
 writeln(tty,' rubrique resultat');
```

```
  writeln(tty);
  writeln(tty,'        ********');
  writeln(tty,'*               *');
  writeln(tty,'*     CIA       *');
  writeln(tty,'*               *');
  writeln(tty,'        ********');
  writeln(tty);
end.
```

ANNEXE II

SPÉCIFICATIONS DES PROCÉDURES DU PROGRAMME

Procédure listsomval.

Soit somval, un record composé de som : un sommet
val, valaj : des réels
dvt (pointeur vers l'avant) : pointeur
arri (pointeur vers l'arrière) : pointeur

Soit dpt (pointeur de départ de la liste), arvl (pointeur s/ le dernier elmt de la liste), lavancer, lreculer, passé, l: des pointeurs

Soit valeurexiste : un entier
Soit vv : un sommet
Soit prm : un caractère
Soit n : entier compris entre 0 et 7
Soit valinit: un réel

Si n vaut 0,
la procédure initialise les pointeurs dpt, arvl, lreculer, lavancer, passé et futur à nil, et place la valeur '0' dans la variable prm.

Si n vaut 1,
la procédure crée un nouvel élément (somval)
tel que som prend la valeur contenue dans vv
val prend la valeur contenue dans valinit
où valinit est acquis interactivement si valeurexiste = '0'

valaj prend la valeur nulle
arri prend la valeur arvl
dvt prend la valeur nil

De plus, si prm valait '0' la procédure modifie dpt qui prend la valeur adresse du nouvel élément et prm devient 'n'
et modifie arvl qui prend également la valeur adresse du nouvel élement, dans tous les cas.

Si n vaut 2, la procédure modifie

- la valeur du pointeur dvt de l'élément précédent arvl dans la liste, qui prend la valeur nil
- la valeur de arvl qui devient la valeur du pointeur arri du dernier élément de la liste (arvl) et efface ce dernier élément.

Si n vaut 3, la procédure modifie
la valeur de l et de lavancer qui prennent la valeur du pointeur dpt.

Si n vaut 4, la procédure modifie
la valeur de l et de lreculer qui prennent la valeur du pointeur arvl.

Si n vaut 5, la procédure modifie
la valeur de l qui prend la valeur du pointeur dvt de l'élément placé en lavancer.

Si n vaut 6, la procédure modifie
la valeur de l qui prend la valeur du pointeur arri de l'élément placé en lreculer.

Si n vaut 7, la procédure efface l'élément pointé par l,
et met à jour les pointeurs arrière et avant de l'élement précédent et suivant, respectivement, pour autant que ceux-ci existent.
Si l'élément l n'a pas de précédent et/ou de suivant, la procédure met à jour le pointeur dpt et/ou arvl respectivement.
Plus précisément, la procédure efface l'élément d'adresse l, la procédure modifie la valeur du pointeur arri de l'élément suivant dans la liste, qui prend la valeur du pointeur arri de l'élement d'adresse l, pour autant qu'il y ait un suivant.

Si cet élément n'existe pas (l désigne le dernier élément de la liste) alors la procédure modifie la valeur du pointeur arvl qui prend la valeur du pointeur arri de l'élément pointé par arvl.

La procédure modifie la valeur du pointeur dvt de l'élément précédent dans la liste, qui prend la valeur du pointeur dvt de l'élément d'adresse l, pour autant qu'il y ait un élément précédent.
Si cet élément n'existe pas (l désigne le premier élément de la liste), alors la procédure modifie la valeur du pointeur dpt qui prend la valeur du pointeur dvt de l'élément pointé par dpt.

Procédure rechvalsom.

Soit XX un sommet
Soit listsomval, une liste de sommets

et l un pointeur

la procédure renvoie une valeur de pas nulle si le sommet XX ne se trouve pas dans la liste somval, et une valeur de pas = 1 si le sommet XX s'y trouve. l pointe alors vers l'élément dont som est XX.

Procédure validsommet.

Si n vaut 1 : soit début un sommet
la procédure renvoie une valeur de correct = '1'
et un message d'erreur approprié si début n'a pas de nom.

Si n vaut 2 : soit fin et début, deux sommets,
la procédure renvoie une valeur de correct = '1'
et un message d'erreur approprié si fin n'a pas de nom
ou si fin = début.

Si n vaut 3 : soit testé, un sommet
soit message, un caractère
soit somval, une liste de sommets
la procédure renvoie une valeur de correct 'n'
si testé ne se trouve pas dans la liste somval
ainsi qu'un message d'erreur adéquat
à condition que la valeur de message soit '0'.

Procédure validtemps.

Soit time, limdel deux entiers
la procédure renvoie une valeur de correct = '1' et un
et un message d'erreur approprié si time est négatif
ou si time est supérieur à limdel.

Procédure validmesure.

Soit coéff, un réel
Soit liminfmes et limsupmes deux entiers, liminfmes $\leqslant$ limsupmes
la procédure renvoie une valeur de correct = '1'
ainsi qu'un message d'erreur si coéff n'est pas compris
entre liminfmes et limsupmes.

Procédure listarcnul.

Soit arcnul un record composé de alpha, oméga : deux sommets
numéro : un entier
avant, arrière : deux pointeurs

Soit p, départ, arrivée, pavancer, preculer, hier, demain : des pointeurs.

Soit premier, un caractère
Soit n, un entier compris entre 0 et 7
Soit nbarcnul, un entier
Soit début, fin, deux sommets

Si n vaut 0,
la procédure initialise les pointeurs départ, arrivée, pavancer, preculer à la valeur nil, et place la valeur '0' dans la variable premier.

Si n vaut 1,
la procédure crée un nouvel élément (arcnul)
tel que alpha prend la valeur contenue dans début
oméga prend la valeur contenue dans fin
numéro " " " de nbarcnul
arrière " " " arrivée
avant " " " nil

De plus, la procédure modifie départ: = adresse du nouvel élément
et premier: = '1'
si premier valait '0'

et la procédure modifie arrivée: = adresse du nouvel élément
dans tous les cas.

Si n vaut 2,
la procédure modifie - la valeur du pointeur avant de l'élement précédent arrivée dans la liste, qui prend la valeur nil
- la valeur de arrivée qui prend la valeur du pointeur arrière du dernier élément de la liste
et efface ce dernier élément.

Si n vaut 3,

la procédure modifie la valeur de p et de pavancer qui prennent la valeur du pointeur départ.

Si n vaut 4,

la procédure modifie la valeur de p et de preculer qui prennent la valeur du pointeur arrivée.

Si n vaut 5,

la procédure modifie la valeur de p qui prend la valeur du pointeur avant de l'élément placé en pavancer.

Si n vaut 6,

la procédure modifie la valeur de p qui prend la valeur du pointeur arrière de l'élément placé en preculer.

Si n vaut 7,

la procédure efface l'élément placé en p,
la procédure modifie la valeur du pointeur arrière de l'élément suivant p dans la liste, qui prend la valeur du pointeur arrière de l'élément d'adresse p, pour autant qu'il y ait un suivant.
Si cet élément n'existe pas (p désigne le dernier élément de la liste), alors la procédure modifie la valeur du pointeur arrivée qui prend la valeur du pointeur arrière de l'élément pointé par arrivée.

La procédure modifie la valeur du pointeur avant de l'élement précédent p, dans la liste, qui prend la valeur du pointeur avant de l'élément d'adresse p, pour autant qu'il y ait un précédent.
Si cet élément n'existe pas (p désigne le premier élément de la liste), alors la procédure modifie la valeur du pointeur départ qui prend la valeur du pointeur avant de l'élément pointé par départ.

Procédure help.

Soit n un entier compris entre 1 et 17

Si n vaut 1,

la procédure fait paraître à l'écran un texte expliquant la méthode d'introduction des données.

Si n vaut 2,

la procédure fait paraître à l'écran un texte d'aide à l'introduction de l'extrémité d'une flèche.

Si n vaut 3,

les explications portent sur l'introduction du cvd, pour la simulation.

Si n vaut 4,

le texte explique les caractéristiques de l'introduction d'une nouvelle flèche dans le schéma.

Si n vaut 5,

le texte fournit des renseignements sur l'introduction de l'extrémité d'une flèche à ajouter.

Si n vaut 6,

le texte fournit des explications quant au retrait d'une flèche du schéma.

Si n vaut 7,

le texte fournit des renseignements sur l'introduction de l'extrémité d'une flèche à retirer.

Si n vaut 8,

le texte fournit des explications quant à la modification d'une flèche du schéma.

Si n vaut 9,

le texte fournit des renseignements sur l'introduction de l'extrémité d'une flèche à modifier.

Si n vaut 10,

le texte présente l'éventail des possibilités de modifications du schéma donné.

Si n vaut 11,

le texte présente une explication de la possibilité de réexécuter la simulation sur le même schéma.

Si n vaut 12,

le texte présente et commente les possibilités de réponse à la question, 'voulez-vous exécuter le programme sur le schéma présenté en dernier lieu'?

Si n vaut 13,

le texte fournit des explications quant au retrait d'un sommet du schéma.

Si n vaut 14,

le texte fournit des explications quant à la modification de la valeur d'un sommet.

Si n vaut 15,

le texte présente et commente les possibilités de réponse à la question 'voulez-vous voir les résultats de vos modifications ?'

Si n vaut 16,

le texte présente et commente les possibilités de réponse à la question 'voulez-vous imprimer les résultats de la simulation ?'

Si n vaut 17,

le texte présente et commente les possibilités de réponse à la question 'voulez-vous voir le détail des opérations qui ont permis d'établir les résultats de la simulation ?'

Procédure introduction

La procédure constitue interactivement un fichier séquentiel de flèches tel que les flèches de même origine sont groupées, représentant un schéma économique, ainsi qu'une liste des arcs de délai nul et une liste des sommets du schéma avec leur valeur respective. La procédure renvoie une valeur d'arrêt = '0' lorsque le schéma est totalement constitué.

Procédure affichage

Soit schéma, un fichier de flèches, séquentiel
Soit somval, une liste des sommets du graphe et leurs valeurs

la procédure affiche à l'écran les flèches du schéma, ainsi que la liste des sommets du schéma et leur valeur initiale.

Procédure recopiedau

Cette procédure recopie le contenu du fichier schéma2 dans le fichier (schéma) où schéma et schéma2 sont des fichiers séquentiels de flèches.

Procédure ajflèche

Soit schéma, un fichier séquentiel de flèches dont les flèches de même origine sont groupées

la procédure acquiert interactivement un nouvel article et renvoie un fichier dont les articles de même origine sont groupés, identiques à schéma, et contenant 0 ou 1 nouvel article (selon que le nouvel article est validé ou pas) ou retourne un message d'erreur si le nouvel article existe déjà, auquel cas schéma est inchangé.
De plus, la procédure met à jour les listes arcnul et somval.

Procédure retflèche.

Soit schéma un fichier séquentiel de flèches,

la procédure, après avoir acquis interactivement la flèche à retirer, renvoie un fichier séquentiel de flèches identiques à celui entré mais dont 0 ou 1 article aura été retiré (selon que l'article à retirer est validé ou pas).
De plus, la procédure met à jour la liste des arcnuls.

Procédure retsom.

Soit schéma un fichier de flèches et
Soit somval une liste de sommets

la procédure obtient interactivement le nom d'un sommet à retirer, et renvoie un message d'erreur si ce sommet n'appartient pas à la liste somval et modifie la liste somval en y retirant ce sommet s'il s'y trouve.
La procédure retire également du fichier schéma les arcs dont l'origine ou l'extrémité égale ce sommet et met à jour, conséquemment, la liste des arcs nuls.

Procédure modflèche.

Soit schéma un fichier séquentiel de flèches,

la procédure acquiert interactivement la flèche à retirer et renvoie un message d'erreurs si la flèche n'existe pas dans le schéma, sinon renvoie le fichier schéma dans lequel 0 ou 1 article aura été modifié.
De plus, la procédure met à jour la liste des arcs nuls.

Procédure modsom.

Soit somval une liste de sommets et de leurs valeurs respectives,

la procédure acquiert interactivement le nom du sommet dont la valeur doit être modifiée, et renvoie un message d'erreur si ce sommet n'appartient pas à la liste, et sinon modifie la valeur de ce sommet dans la liste.

Procédure modification.

La procédure acquiert interactivement le code correspondant à la modification à effectuer sur le schéma et la réalise.

Procédure demaffich.

La procédure acquiert interactivement la réponse de l'utilisateur à la question 'voulez-vous voir les résultats de vos modifications ?'
et renvoie - un message indicateur si la réponse est help
- un message d'erreur si la réponse est ≠ 'o', 'n', 'help'
- les flèches du schéma et la liste des sommets avec leur valeur initiale si la réponse est 'o'

et n'a aucun effet si la réponse est non.

Procédure demandemodification.

La procédure acquiert interactivement la réponse de l'utilisateur à la question 'voulez-vous modifier votre schéma ?'

et renvoie - un message indicateur si la réponse est 'help'
- un message d'erreur si la réponse n'est ni 'o', ni 'n', ni 'help'.

Si la réponse est oui,

la procédure propose diverses modifications et les exécute selon le gré de l'utilisateur. De plus, la procédure propose l'affichage des résultats des modifications et l'effectue selon le gré de l'utilisateur.

La procédure n'a aucun effet si la réponse est non.

Procédure gestmat.

Soit lignecol un record composé de nom : un sommet
élast: un réel
noligne, nocol: 2 entiers
avligne, avcol: 2 pointeurs

Soit s, pcdtcol, pcdtligne, somcvd : des pointeurs

Soit cvd un sommet
tho un réel
tamp une flèche
I un entier
et C(0:TI) un vecteur de nombres entiers

Soit n un entier compris entre 0 et 6

Si n vaut 0,
la procédure initialise les pointeurs somcvd, pcdtcol, pcdtligne à la valeur nil

Si n vaut 1,
la procédure crée un nouvel élément (lignecol)
tel que nom prend la valeur de cvd
élast " " " 1
noligne " " " 0
nocol " " " 1
avligne " " " nil
avcol " " " nil

de plus, la procédure modifie la valeur du pointeur pcdtcol et pcdtligne qui prennent la valeur adresse du nouvel élément
et la valeur du pointeur somcvd qui prend cette même valeur.

Si n vaut 2,

la procédure crée un nouvel élément (lignecol)
tel que nom prend la valeur de tamp.ext

élast	"	"	"	tho x tamp.taux
noligne	"	"	"	I
nocol	"	"	"	1
avligne	"	"	"	nil
avcol	"	"	"	nil

De plus, la procédure modifie

- le pointeur avcol de l'élément pointé par pcdtcol, qui prend la valeur adresse de ce nouvel élement

- la valeur du pointeur pcdtcol et pcdtligne qui prennent la valeur adresse du nouvel élement.

Si n vaut 3,

la procédure crée un nouvel élement (lignecol)
tel que nom prend la valeur tamp.ext

élast	"	"	"	tho x tamp.ext
noligne	"	"	"	I
nocol	"	"	"	C(I)
avligne	"	"	"	nil
avcol	"	"	"	nil

De plus, la procédure modifie

- le pointeur avligne de l'élement pointé par pcdtligne, qui prend la valeur adresse du nouvel élément

- le pointeur pcdtligne qui prend également cette valeur.

Si n vaut 4,

la procédure modifie s qui prend la valeur du pointeur avcol de l'élément pointé par s, pour autant que ce pointeur ne soit pas nil.

narc' = $\sum_{\mu=1}^{n'}$ nombre d'arcs de délai i' issus de X_μ

nchem = $\sum_{\mu=1}^{n'}$ nombre de chemins différents, de délai total nul, issus de X_μ

L'exécution de la boucle avec $i = i' > 0$, I = I',

C(I' - i') = n', C(I') = m',

et (D(I' - i', 1)),....., D(I' - i', n') = (X_1, ..., $X_{n'}$)

- établit C(I') = m' + narc'
- place sur D(I', m' + 1), ... D(I', m' + narc') les extrémités des arcs issus de X_1, ..., $X_{n'}$ de délai i'
- ne modifie pas les valeurs de i et I, ni des autres éléments de C et D.

L'exécution de la boucle avec $i' = 0$, I = I', c(I') = m' et
(D(I', 1), ..., D(I', m')) = (X_1, ..., $X_{m'}$)

- établit C(I') = m' + nchem
- place sur D(I', m' + 1), ..., D(I', m' + nchem) les extrémités des chemins différents de délai total nul issus de X_1, ..., $X_{m'}$
- ne modifie pas les valeurs de I, i ni les autres éléments de C et D.

Procédure Dboucle

Soit E_μ = l'ensemble des sommets du graphe, extrémités de chemins différents de délai total μ issus du cvd

Soit I', entier tel que $0 \leqslant I' \leqslant TI$

Si n vaut 5,

la procédure modifie s qui prend la valeur du pointeur avligne de l'élément pointé par s, pour autant que le pointeur ne soit pas nil.

Si n vaut 6,

la procédure modifie s qui prend la valeur du pointeur somcvd.

Procédure positionner (l,c)

la procédure modifie la valeur du pointeur s qui prend la valeur adresse de l'élément dont noligne = l et dont locol = c.

Procédure Qboucle

Soit X', un sommet du graphe,
Soit I', un entier tel que $0 \leqslant I' \leqslant TI$
et i', un entier quelconque
Soit C', nombre entier $\geqslant 0$
nb', le nombre d'arcs de délai i' issus de X'

L'exécution de la boucle avec X=X', i=i', I=I' et C'=C(I')
- établit C(I'): = C' + nb'
- place sur D(I', C'+1) ...D(I', C'+nb') les extrémités des arcs de délai i' issus de X'
- ne modifie pas les valeurs de i et I, ni les valeurs des autres éléments de C et de D.

Procédure Tboucle

Soit I' entier tel que $0 \leqslant I' \leqslant TI$
i' entier tel que $0 \leqslant i' \leqslant I'$
m', n' entiers $\geqslant 0$
X_1, ..., $X_{n'}$, suite de sommets du graphe

si, initialement - chacun des éléments C(1), ..., C(I' - 1) a une valeur
- $D(\mu, 1), \ldots\ldots, D(\mu, C(\mu)) = E_\mu$ pour $0 \leqslant \mu \leqslant I'$
- $I = I'$

l'exécution de la boucle
- affecte une valeur $\geqslant 0$ à C(I') = nombre de cvd-chemins différents de délai total I',
- établit $D(I, 1), \ldots\ldots, D(I', C(I')) = E_{I'}$
- ne modifie pas C(1), ..., C(I' - 1), ni les lignes D(1), ..., D(I' - 1), ni la valeur de I.

Procédure Pboucle

L'exécution de la boucle - affecte une valeur $\geqslant 0$ à C(0), ..., C(TI), avec TI donné fini

et - établit $D(\mu,1), \ldots, D(\mu, C(\mu)) = E_\mu$

$\forall \mu = 0 \leqslant \mu \leqslant TI$.

Procédure resultaffich

Soit imp, un réel
Soit gestmat, une liste d'éléments lignecol

la procédure parcourt les éléments de la liste gestmat et, pour chaque élément, affiche à l'écran les valeurs de nom, elast, imp x elast ainsi que la valeur initiale (val) des sommets dont som = nom et la valeur de val + (imp x elast).

Procédure recapaffich

La procédure affiche à l'écran les résultats de la simulation en récapitulant pour chaque sommet les influences qu'il a subies dans un même délai.

Procédure schemaffich

Soit résultat, un fichier texte
Soit schéma, un fichier séquentiel de flèches
Soit somval, une liste des sommets du schéma et leur valeur initiale,

la procédure affiche, dans le fichier résultat, à la suite de ce qui s'y trouve déjà, les flèches du schéma, ainsi que la liste des sommets de somval et leur valeur initiale.

Procedure printdetail

Soit imp, un réel
Soit gestmat, une liste d'éléments lignecol,

la procédure parcourt les éléments de la liste gestmat et, pour chaque élément, affiche dans le fichier résultat, à la suite de ce qui s'y trouve déjà, les valeurs de nom, elast, imp x elast, ainsi que la valeur initiale (val) du sommet dont som = nom et la valeur de val + (imp x elast).

Procédure printdemande

Soit résultat, un fichier texte,

la procédure acquiert interactivement la réponse de l'utilisateur à la question 'voulez-vous imprimer les résultats de cette simulation?'

et renvoie - un message indicateur si la réponse est 'help'
- un message d'erreur si la réponse est différente de 'o', 'n', 'help'

affiche les résultats dans le fichier 'résultat', à la suite de ce qui s'y trouve déjà, si la réponse est 'o'

n'effectue rien si la réponse est 'n'.

Procédure printresultat

La procédure affiche dans le fichier résultat, à la suite de ce qui s'y trouve déjà, les résultats de la simulation en récapitulant pour chaque sommet les influences qu'il a subies dans un même délai.

De plus, la procédure acquiert interactivement la réponse à la question 'désirez-vous l'impression des détails des résultats ?'

et renvoie un message d'erreur si la réponse est différente de 'o', 'n' ou affiche dans le fichier résultat, à la suite de ce qui s'y trouve déjà, les détails si la réponse est 'o'.

Procédure mémoire

Soit liminfmes et limsupmes, deux entiers,

la procédure acquiert interactivement le nom du cvd et le valide comme étant un sommet du schéma, le temps imparti et le valide comme étant compris entre liminfmes et limsupmes, l'impulsion de départ.
Puis, la procédure effectue la simulation et propose l'affichage des résultats récapitulatifs et détaillés, et l'effectue selon le gré de l'utilisateur.

Procédure listcircnul

Soit lélément, un record composé de elmt : un sommet
num : un entier
lavant, larrière : deux pointeurs

Soit e, ledépart, larrivée, eavancer, ereculer : des pointeurs
Soit p, pointeur vers un arcnul
u, un entier
et lepremier, un caractère
Soit n, un entier compris entre 0 et 6

Si n vaut 0,

la procédure initialise les pointeurs ledépart, larrivée, eavancer, ereculer à la valeur nil, et donne à lepremier la valeur 'o'.

Si n vaut 1,

la procédure crée un nouveau lélément
tel que elmt prend la valeur alpha de l'arcnul pointé par p
num " " " u
lavant " " " nil
larrière " " " larrivée

De plus, si lepremier = 'o', alors la procédure modifie ledépart qui prend la valeur adresse du nouvel élément et lepremier prend la valeur 'n'.
Dans tous les cas, larrivée prend la valeur adresse du nouvel élément.

Si n vaut 2,

la procédure modifie
- la valeur du pointeur lavant de l'élément précédent larrivée sur la liste, qui prend la valeur nil
- la valeur de larrivée qui prend la valeur du pointeur larrière du dernier élément de la liste

et la procédure efface le dernier élément.

Si n vaut 3,

la procédure modifie la valeur de e et de eavancer qui prennent la valeur du pointeur de départ.

Si n vaut 4,

la procédure modifie la valeur de e et de ereculer qui prennent la valeur du pointeur larrivée.

Si n vaut 5,

la procédure modifie la valeur de e qui prend la valeur du pointeur lavant de l'élément placé en eavancer.

Si n vaut 6,

la procédure modifie la valeur de e qui prend la valeur du pointeur larrière de l'élément placé en ereculer.

Procédure gestelmt

Soit elemcirc, un record composé de mot : un sommet
nbe : un entier
av, ar : deux pointeurs

Soit prem, un caractère
Soit e, un pointeur vers un lélément
k, un entier
Soit c, dep, arriv, cavancer, creculer : des pointeurs
Soit n, un entier compris entre 0 et 6

Si n vaut 0,

la procédure initialise les pointeurs dep, arriv, cavancer, creculer à nil et prem prend la valeur 'o'.

Si n vaut 1,

la procédure crée un nouvel élément (elemcirc)
tel que mot prend la valeur elmt de lélément désigné par c
nbe " " " k
av " " " nil
ar " " " arriv

De plus, si prem = 'o', alors la procédure modifie dep qui prend la valeur adresse du nouvel élément et prem prend la valeur 'n'.

Dans tous les cas, la procédure modifie arriv qui prend la valeur adresse du nouvel élément.

Si n vaut 2,

la procédure modifie
- la valeur du pointeur av de l'élément précédent arriv sur la liste, qui prend la valeur nil
- la valeur de arriv qui prend comme nouvelle valeur celle du pointeur ar du dernier élément de la liste

la procédure efface ce dernier élément.

Si n vaut 3,
la procédure modifie la valeur de c et de cavancer qui prennent la valeur du pointeur dep.

Si n vaut 4,

la procédure modifie la valeur de c et de creculer qui prennent la valeur du pointeur arriv.

Si n vaut 5,

la procédure modifie la valeur de c qui prend la valeur du pointeur av de l'élément placé en cavancer.

Si n vaut 6,

la procédure modifie la valeur de c qui prend la valeur du pointeur ar de l'élément placé en creculer.

Procédure gestmarq

Soit marquage, un record composé de mark, number : deux entiers
forw, back : des pointeurs
Soit first, un caractère
Soit m, departure, arrival, mavancer, mreculer : des pointeurs
Soit v, w, des entiers
Soit n, un entier compris entre 0 et 6.

Si n vaut 0,

la procédure initialise les pointeurs departure, arrival, mavancer, mreculer à la valeur nil et first prend la valeur 'o'.

Si n vaut 1,

la procédure crée un nouvel élément (marquage)
tel que mark prend la valeur v
number " " w
forw " " nil
back " " arrival

De plus, si first = 'o', alors la procédure modifie first qui devient 'n' et modifie departure qui prend la valeur adresse du nouvel élément. Dans tous les cas, arrival prendra cette même valeur.

Si n vaut 2,

la procédure modifie
- la valeur du pointeur forw de l'élement précédent arrival sur la liste,qui prend la valeur nil

- la valeur de arrival qui prend comme nouvelle valeur celle du pointeur back du dernier élément de la liste.

La procédure efface ce dernier élément.

Si n vaut 3,

la procédure modifie la valeur de m et de mavancer qui prennent la valeur du pointeur départure.

Si n vaut 4,

la procédure modifie la valeur de m et de mreculer qui prennent la valeur du pointeur arrival.

Si n vaut 5,

la procédure modifie la valeur de m qui prend la valeur du pointeur forw de l'élément placé en mavancer.

Si n vaut 6,

la procédure modifie la valeur de m qui prend la valeur du pointeur back de l'élément placé en mreculer.

Procédure comparaison

La procédure compare le dernier circuit à tous ceux de même longueur qui se trouvent dans la liste gestelmt, jusqu'à en trouver un qui comporte les mêmes sommets dans le même ordre, auquel cas la procédure renvoie une valeur de comp égale à '+'.
Si la procédure ne trouve pas de tel circuit, elle renvoie une valeur de comp égale à '-'.

Procédure effacement

Soit w, un entier

la procédure efface les sommets du dernier circuit de la liste gestelmt, décrémente w d'une unité, et efface le dernier élément de la liste gestmarq.

Procédure ecritcircnul

La procédure affiche à l'écran un message indiquant à l'utilisateur la présence d'un circuit de délai total nul et donne les sommets qui le composent. De plus, la procédure modifie la valeur de indiccirc qui prend la valeur 1.

Procédure vérifcirc

La procédure vérifie s'il existe un circuit dans la liste listcircnul, auquel cas elle le place dans la liste gestelmt. Elle le compare aux circuits de même longueur existant déjà dans cette liste et efface le nouveau circuit si la comparaison s'avère positive.
Sinon,elle l'imprime à l'écran.

Procédure explore

Soit p, un pointeur vers un arcnul,

la procédure explore, en profondeur d'abord, les descendants du sommet oméga de p et les place dans la liste listcircnul. Elle vérifie l'existence d'un circuit parmi les éléments de listcircnul, auquel cas elle le place dans gestelmt où elle le compare aux circuits de même longueur existant déjà dans cette liste.
Si la comparaison est positive, il y a effacement du dernier circuit de cette liste. Sinon, il y a affichage à l'écran de ce circuit.

Procédure rechcircnul

La procédure place successivement les éléments de la liste des arcs nuls dans le premier élément de la liste circnul et effectue une recherche de circuits parmi les descendants de cet élément.

La procédure renvoie une valeur de indiccirc = 1 si elle détecte la présence d'un circuit.

Procédure exécution

Soit schéma, un fichier de flèches,
Soit nart, un entier $\geqslant 0$,

la procédure renvoie les résultats 0,1 ou plusieurs simulations sur ce schéma et renvoie un message d'erreur si nart = 0 ou si le schéma comporte un circuit de délai total nul.
Dans ce dernier cas, le message donnera les éléments constituant ce circuit.

Procédure demandeexécution

La procédure acquiert interactivement la réponse à la question 'voulez-vous exécuter le programme avec le schéma dernièrement entré ?'

et renvoie - un message indicateur si la réponse est 'help'
- un message d'erreur si la réponse est différente de 'o', 'n', 'help'.

Si la réponse est 'o',

la procédure effectue une vérification des conditions d'exécution de la simulation et si ces conditions sont remplies, elle effectue la simulation. Sinon, elle renvoie un message d'erreur.

Procédure reformation

Soit schéma, un fichier non vide, séquentiel de flèches,
Soit somval, la liste des sommets et de leur valeur initiale,

la procédure reconstitue la liste des arcs nuls, recalcule nbarcnul et nart, c'est-à-dire les incrémente d'une unité à chaque flèche de délai nul trouvée dans schéma et à chaque flèche respectivement.

La procédure acquiert interactivement les valeurs des sommets se trouvant dans le schéma, mais qui ne seraient pas dans la liste des sommets.

Procédure sauvetage

Soit somval, une liste de sommets et leur valeur initiale,

la procédure recopie la liste somval des sommets et de leur valeur initiale dans le fichier (fsommet).

Procédure réhabilitation

Soit fsommet, un fichier séquentiel de sommets et de leur valeur respective,

la procédure reconstruit la liste somval des sommets à partir du fichier (fsommet).

BIBLIOGRAPHIE

- Algèbre moderne et théorie des graphes,

  B. Roy
  Dunod, 1969 - 1970

- Graphes et algorithmes,

  M. Gondran et M. Minoux
  Eyrolles, Paris 1972

- The Design and Analysis of Computer Algorithms,

  A.V. Aho, J.E. Hopcroft et J.D. Ullman
  Addison - Vivlag, Reading Mass. 1970

- On the criteria to be used in decomposing systems into modules,

  D.L. Parnas
  CACM, December 72

- A technique for software module specification with examples,

  D.L. Parnas
  CACM 15, May 72

- Principles of Package design,

  B. Meyer
  CACM, July 1982 Volume 25, number 7

- Principles of program design,

  M.A. Jackson
  Academic Press, New-York 1975

- Rapport de séminaire d'économie publique (troisième phase)

  C. Debatty, C. Ledouble
  Année académique 82-83